(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号
特許第6357807号
(P6357807)
(45)発行日　平成30年7月18日(2018.7.18)　(24)登録日　平成30年6月29日(2018.6.29)

(51)Int.Cl.　F I
G06F 9/50 (2006.01)　G06F 9/50 15OE

請求項の数 9　(全 25 頁)

(21)出願番号　特願2014-43333(P2014-43333)
(22)出願日　平成26年3月5日(2014.3.5)
(65)公開番号　特開2015-170054(P2015-170054A)
(43)公開日　平成27年9月28日(2015.9.28)
審査請求日　平成28年11月2日(2016.11.2)

(73)特許権者　000005223
富士通株式会社
神奈川県川崎市中原区上小田中４丁目１番１号
(74)代理人　100089118
弁理士　酒井　宏明
(72)発明者　上田　晴康
神奈川県川崎市中原区上小田中４丁目１番１号　富士通株式会社内
(72)発明者　松田　雄一
神奈川県川崎市中原区上小田中４丁目１番１号　富士通株式会社内
(72)発明者　前田　高光
神奈川県川崎市中原区上小田中４丁目１番１号　富士通株式会社内
最終頁に続く

(54)【発明の名称】タスク割当プログラム、タスク実行プログラム、マスタサーバ、スレーブサーバおよびタスク割当方法

(57)【特許請求の範囲】
【請求項１】
複数のMap処理タスクを含むMap処理および複数のReduce処理タスクを含むReduce処理を含むMapReduce処理の各タスクを複数のスレーブサーバに割当てるマスタサーバに、
前記複数のスレーブサーバ夫々に、前記複数のMap処理タスクを割り当て、
前記複数のMap処理タスクの全部が完了する前に、対象となる複数のMap処理タスクの実行結果を用いる前記複数のReduce処理タスク夫々が処理対象とするデータ量を、処理が完了した各Map処理タスクから送信される実行結果に含まれるデータ量に基づいて予測し、
前記複数のReduce処理タスクを前記複数のスレーブサーバ夫々に割当てる際に、予測された前記データ量が閾値を超えるReduce処理タスクである高負荷Reduce処理タスクを割当てるスレーブサーバに、前記高負荷Reduce処理タスクの分散処理を指示する、
処理を実行させることを特徴とするタスク割当プログラム。
【請求項２】
前記予測する処理は、全Map処理タスクにおける所定の割合の処理が終了した場合、または、最初のMap処理タスクが終了してから所定時間が経過した場合に、前記複数のReduce処理タスク夫々のデータ量を予測することを特徴とする請求項１に記載のタスク割当プログラム。
【請求項３】
複数のMap処理タスクを含むMap処理および複数のReduce処理タスクを含むReduce処理を含むMapReduce処理の各タスクを実行するスレーブサーバに、

前記各タスクの割当てを実行するマスタサーバからReduce処理タスクが割当てられた場合に、当該Reduce処理タスクが分散処理の実行指示が付された高負荷Reduce処理タスクであるか否かを判定し、
前記高負荷Reduce処理タスクであると判定された場合、前記高負荷Reduce処理タスクを分散処理で実行し、前記高負荷Reduce処理タスクではない通常のReduce処理タスクと判定された場合、前記通常のReduce処理タスクを分散させずに実行する
処理を実行させることを特徴とするタスク実行プログラム。
【請求項４】
前記マスタサーバから割り当てられた前記Map処理タスクの実行が完了した場合に、完了した前記Map処理タスクの処理結果を用いるReduce処理タスクに関するデータ量を含む完了結果を前記マスタサーバに送信する処理を、前記スレーブサーバにさらに実行させることを特徴とする請求項３に記載のタスク実行プログラム。
【請求項５】
前記実行する処理は、前記スレーブサーバが有するプロセッサの数、ディスクの数、予め指定された数の少なくとも１つを用いて、前記高負荷Reduce処理タスクをサブタスクに分割して、複数のプロセッサで並列実行させることを特徴とする請求項３または４に記載のタスク実行プログラム。
【請求項６】
前記実行する処理は、前記高負荷Reduce処理タスクではない通常のReduce処理タスクの実行中に前記高負荷Reduce処理タスクが割当てられた場合、前記通常のReduce処理タスクの実行を中止して、前記高負荷Reduce処理タスクを前記サブタスクに分割して実行することを特徴とする請求項５に記載のタスク実行プログラム。
【請求項７】
複数のMap処理タスクを含むMap処理および複数のReduce処理タスクを含むReduce処理を含むMapReduce処理の各タスクを複数のスレーブサーバに割当てるマスタサーバにおいて、
前記複数のスレーブサーバ夫々に、前記複数のMap処理タスクを割り当てる割当部と、
前記複数のMap処理タスクの全部が完了する前に、対象となる複数のMap処理タスクの実行結果を用いる前記複数のReduce処理タスク夫々が処理対象とするデータ量を、処理が完了した各Map処理タスクから送信される実行結果に含まれるデータ量に基づいて予測する予測部と、
前記複数のReduce処理タスクを前記複数のスレーブサーバ夫々に割当てる際に、予測された前記データ量が閾値を超えるReduce処理タスクである高負荷Reduce処理タスクを割当てるスレーブサーバに、前記高負荷Reduce処理タスクの分散処理を指示する送信部と
を有することを特徴とするマスタサーバ。
【請求項８】
複数のMap処理タスクを含むMap処理および複数のReduce処理タスクを含むReduce処理を含むMapReduce処理の各タスクを実行するスレーブサーバにおいて、
前記各タスクの割当てを実行するマスタサーバからReduce処理タスクが割当てられた場合に、当該Reduce処理タスクが分散処理の実行指示が付された高負荷Reduce処理タスクであるか否かを判定する判定部と、
前記高負荷Reduce処理タスクであると判定された場合、前記高負荷Reduce処理タスクを分散処理で実行し、前記高負荷Reduce処理タスクではない通常のReduce処理タスクと判定された場合、前記通常のReduce処理タスクを分散させずに実行する処理実行部と
を有することを特徴とするスレーブサーバ。
【請求項９】
複数のMap処理タスクを含むMap処理および複数のReduce処理タスクを含むReduce処理を含むMapReduce処理の各タスクを複数のスレーブサーバに割当てるマスタサーバが、
前記複数のスレーブサーバ夫々に、前記複数のMap処理タスクを割り当て、
前記複数のMap処理タスクの全部が完了する前に、対象となる複数のMap処理タスクの実

行結果を用いる前記複数のReduce処理タスク夫々が処理対象とするデータ量を、処理が完了した各Map処理タスクから送信される実行結果に含まれるデータ量に基づいて予測し、
前記複数のReduce処理タスクを前記複数のスレーブサーバ夫々に割当てる際に、予測された前記データ量が閾値を超えるReduce処理タスクである高負荷Reduce処理タスクを割当てるスレーブサーバに、前記高負荷Reduce処理タスクの分散処理を指示する、
処理を実行することを特徴とするタスク割当方法。

【発明の詳細な説明】

【技術分野】

【０００１】

本発明は、タスク割当プログラム、タスク実行プログラム、タスク割当装置、タスク実行装置およびタスク割当方法に関する。

【背景技術】

【０００２】

クラウドコンピューティングの普及に伴い、クラウド上に保存される大量のデータを複数のサーバで分散して処理を実行する分散処理システムが利用されている。分散処理システムとしては、HDFS（Hadoop Distributed File System）とMapReduce処理とを基盤技術とするHadoop（登録商標）が知られている。

【０００３】

HDFSは、複数のサーバにデータを分散格納するファイルシステムである。MapReduceは、HDFS上のデータをタスクと呼ばれる単位で分散処理する仕組みであり、Map処理、Shuffleソート処理、Reduce処理を実行する。

【０００４】

MapReduceによる分散処理においては、マスタサーバが、ハッシュ関数等を用いて、複数のスレーブサーバに対してMap処理やReduce処理のタスクを割り当てるとともに、分割したデータを各スレーブサーバに送信する。そして、各スレーブサーバが、割り当てられたタスクを実行する。

【０００５】

スレーブサーバに対するタスクの割り当ては、ハッシュ関数等を用いることにより、均等に行われる。その一方で、各Reduceタスクに対応する処理量は、Reduceタスクに対応するキー等に関連づけられたReduce対象のデータ量等により均等になるとは限らない。

【０００６】

Reduceタスクに対応する処理量により、各スレーブサーバでの処理完了時間が異なることから、複数タスクからなるジョブ全体の完了が、最も処理が遅いスレーブサーバの処理完了に左右されることとなる。このため、全Reduceタスクをスレーブサーバに割り当てた後、各Reduceタスクに対応する処理量が均等となるように、データ量を調整する技術が知られている。

【先行技術文献】

【特許文献】

【０００７】

【特許文献１】特開２０１２－１１８６６９号公報

【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、各Reduceタスクの処理量が不均一となることは、入力データやMap処理の結果など様々な影響により異なるので、上記技術のような調整処理を行うことが、ジョブ全体の完了時間を早くするものとは限らない。

【０００９】

例えば、Reduceタスクの処理量を調整するには、全Map処理の完了を待ってから調整することになるので、各スレーブサーバでのReduce処理の実行開始が遅れることになり、却ってジョブ全体の処理時間が長くなる場合もある。

【００１０】
１つの側面では、ジョブ全体の完了時間を短縮することができるタスク割当プログラム、タスク実行プログラム、タスク割当装置、タスク実行装置およびタスク割当方法を提供することを目的とする。
【課題を解決するための手段】
【００１１】
第１の態様では、第１のサーバ装置に、複数の第２のサーバ装置夫々に第１の処理を割り当てる処理を実行させる。第１のサーバ装置に、前記複数の第２のサーバ装置夫々に割り当てる、前記第１の処理タスクの実行結果を用いて実行される第２の処理タスクに関連する前記第１の処理タスクの完了通知を受信した場合に、前記第２の処理タスクの処理量を見積もる処理を実行させる。第１のサーバ装置に、前記第２の処理タスクを割り当てる前記第２のサーバ装置に、見積もった前記処理量に関連した情報を送信する処理を実行させる。
【発明の効果】
【００１２】
１つの側面として、ジョブ全体の完了時間を短縮することができる。
【図面の簡単な説明】
【００１３】
【図１】図１は、実施例１に係る分散処理システムの全体構成例を示す図である。
【図２】図２は、実施例１に係るマスタサーバの機能構成を示す機能ブロック図である。
【図３】図３は、ジョブリストＤＢに記憶される情報の例を示す図である。
【図４】図４は、タスクリストＤＢに記憶される情報の例を示す図である。
【図５】図５は、Map処理の完了通知の例を示す図である。
【図６】図６は、実施例１に係るスレーブサーバの機能構成を示す機能ブロック図である。
【図７】図７は、Map処理を説明する図である。
【図８】図８は、Shuffle処理を説明する図である。
【図９】図９は、Reduce処理を説明する図である。
【図１０】図１０は、Reduce処理タスクの処理量を予測してフラグを設定する処理を説明する図である。
【図１１】図１１は、実施例１に係るマスタサーバが実行する処理の流れを示すフローチャートである。
【図１２】図１２は、マスタサーバが実行する該当タスクの完了処理の流れを示すフローチャートである。
【図１３】図１３は、実施例１に係るスレーブサーバが実行する処理の流れを示すフローチャートである。
【図１４】図１４は、スレーブサーバが実行するReduce処理タスクの起動処理の流れを示すフローチャートである。
【図１５】図１５は、スレーブサーバが実行するReduce処理タスクの分割処理の流れを示すフローチャートである。
【図１６】図１６は、実施例２に係るReduce処理タスクの割当処理の流れを示すフローチャートである。
【図１７】図１７は、実施例３に係るReduce処理タスクの割当処理の流れを示すフローチャートである。
【図１８】図１８は、各サーバのハードウェア構成例を示す図である。
【発明を実施するための形態】
【００１４】
以下に、本願の開示するタスク割当プログラム、タスク実行プログラム、タスク割当装置、タスク実行装置およびタスク割当方法の実施例を図面に基づいて詳細に説明する。なお、この実施例によりこの発明が限定されるものではない。なお、各実施例は、適宜組み

合わせることができる。
【実施例１】
【００１５】
［全体構成］
図１は、実施例１に係る分散処理システムの全体構成例を示す図である。図１に示すように、この分散処理システムは、入出力ＤＢ（DataBase）サーバ２、マスタサーバ１０、複数のスレーブサーバ３０がネットワーク１を介して互いに通信可能に接続される。
【００１６】
この分散処理システムでは、Hadoop（登録商標）などの分散処理フレームワークを使用した分散処理アプリケーションが各計算機で実行されており、データ基盤としてHDFSなどを使用する。
【００１７】
入出力ＤＢサーバ２は、分散処理対象のデータのメタ情報等を記憶するデータベースサーバである。例えば、入出力ＤＢサーバ２が記憶するメタ情報は、どのデータがどのスレーブサーバ３０に格納されているかを特定するのに使用される。
【００１８】
マスタサーバ１０は、分散処理システムを統括的に管理するサーバである。例えば、マスタサーバ１０は、入出力ＤＢサーバ２が記憶するメタ情報から、どのデータがいずれのスレーブサーバ３０に格納されているのかを特定する。また、マスタサーバ１０は、各スレーブサーバ３０に割当てるタスクやジョブなどを管理し、Map処理やReduce処理などのタスクをスレーブサーバ３０に割当てる。
【００１９】
各スレーブサーバ３０は、分散処理アプリケーションを実装し、Map処理やReduce処理を実行して、ＨＤＦＳで管理されるデータを分散処理するサーバである。例えば、スレーブサーバ３０は、複数のプロセッサ、複数のディスクを有する。また、各スレーブサーバ３０は、一意に識別される識別子が割り与えられる。
【００２０】
このスレーブサーバ３０は、入出力ＤＢサーバ２から取得したデータに対して、マスタサーバ１０から割当てられたMap処理のタスクを実行する。また、スレーブサーバ３０は、各スレーブサーバのMap処理結果を用いて、Shuffleソート処理を実行し、マスタサーバ１０から割り当てられたReduce処理のタスクを実行する。
【００２１】
ここで、各処理について説明する。Map処理は、ユーザが定義したMap関数を実行する処理である。例えば、Map処理は、入力データから中間結果として「Key、Value」のペアを出力する。Shuffleソート処理は、Map処理の結果を「Key」でソートし、同じ「Key」を有する「Key、Value」ペアをマージする。Reduce処理は、ユーザが定義したReduce関数を実行する処理である。例えば、Reduce処理は、Shuffleソート処理の結果から、同じ「Key」の「Value」に対して重ね合わせ処理を実行して、新しい形式の「Key、Value」のペアを生成する。
【００２２】
このような状態において、マスタサーバ１０は、複数のスレーブサーバ３０夫々にMap処理を割り当てる。マスタサーバ１０は、複数のスレーブサーバ３０夫々に割り当てる、Map処理タスクの実行結果を用いて実行されるReduce処理タスクに関連するMap処理タスクの完了通知を受信した場合に、Reduce処理タスクの処理量を見積もる。マスタサーバ１０は、Reduce処理タスクを割り当てるスレーブサーバ３０に、見積もった処理量に関連した情報を送信する。
【００２３】
各スレーブサーバ３０は、Map処理タスクの実行結果を用いるReduce処理タスクをスレーブサーバ３０に割り当てるマスタサーバ１０から、Reduce処理タスクに処理量に関連した情報を受信する。各スレーブサーバ３０は、マスタサーバ１０から割り当てられたRedu

ce処理タスクを実行する際に、関連した情報に応じて、Reduce処理タスクの処理方法を変更する。

【００２４】

このように、マスタサーバ１０が、一部のMap処理タスクの終了結果から処理量が多いReduce処理タスクを検出してスレーブサーバ３０に通知するので、スレーブサーバ３０がそのReduce処理タスクを分割して並列処理でき、全処理の完了時間を短縮できる。

【００２５】

［マスタサーバの構成］

図２は、実施例１に係るマスタサーバの機能構成を示す機能ブロック図である。図２に示すように、マスタサーバ１０は、通信制御部１１と、記憶部１２と、制御部１３とを有する。

【００２６】

通信制御部１１は、スレーブサーバ３０などの他装置と通信を実行する処理部であり、例えばネットワークインタフェースカードなどである。例えば、通信制御部１１は、各スレーブサーバ３０に、Map処理タスクやReduce処理タスクを送信する。また、通信制御部１１は、Map処理結果等を各スレーブサーバ３０から受信する。

【００２７】

記憶部１２は、ジョブリストＤＢ１２ａとタスクリストＤＢ１２ｂとを有する記憶装置であり、例えばメモリやハードディスクなどである。また、記憶部１２は、制御部１３が実行するプログラムなどを記憶する。

【００２８】

ジョブリストＤＢ１２ａは、分散処理対象のジョブ情報を記憶するデータベースである。図３は、ジョブリストＤＢ１２ａに記憶される情報の例を示す図である。図３に示すように、ジョブリストＤＢ１２ａは、「JobID、総Mapタスク数、総Reduceタスク数、Reduce割当許可」を対応付けて記憶する。

【００２９】

ここで記憶される「JobID」は、ジョブを識別する識別子である。「総Mapタスク数」は、ジョブに含まれるMap処理タスクの総数である。「総Reduceタスク数」は、ジョブに含まれるReduce処理タスクの総数である。「Reduce割当許可」は、Reduce処理タスクが割当可能な状態か否かを示し、可能な場合は「true」が設定され、可能ではない場合は「false」が設定され、ジョブの新規追加時も「false」が設定される。なお、「JobID、総Mapタスク数、総Reduceタスク数」は、管理者等によって設定更新される。

【００３０】

図３の例では、「JobID」が「１」のジョブは、４つのMap処理タスクと２つのReduce処理タスクで構成され、現在はまだ割当することができない状態であることを示す。同様に、「JobID」が「２」のジョブは、４つのMap処理タスクと２つのReduce処理タスクで構成され、現在はまだ割当することができない状態であることを示す。

【００３１】

タスクリストＤＢ１２ｂは、Map処理タスクやReduce処理タスクに関する情報を記憶するデータベースである。図４は、タスクリストＤＢに記憶される情報の例を示す図である。図４に示すように、タスクリストＤＢ１２ｂは、「JobID、TaskID、種別、Reduceの項番、データのあるスレーブID、状態、割り当てスレーブID、必要スロット数、処理データ量、フラグ」を記憶する。

【００３２】

ここで記憶される「JobID」は、ジョブを識別する識別子である。「TaskID」は、タスクを識別する識別子である。「種別」は、Map処理やReduce処理を示す情報である。「データのあるスレーブID」は、Map処理対象のデータを保持するスレーブサーバ３０を識別する識別子であり、例えばホスト名などである。「状態」は、該当タスクが処理完了（Done）状態、実行中（Running）、割り当て前（Not assigned）のいずれであるかを示す。「Reduceの項番」は、該当Reduceの実行順を示す。「割り当てスレーブID」は、タスクが

割当てられたスレーブサーバ３０を識別する識別子であり、例えばホスト名などである。「必要スロット数」は、タスクを実行するのに使用するスロット数である。「処理データ量」は、該当Reduce処理タスクのデータ量である。「フラグ」は、該当Reduce処理タスクの処理方法の変更を指示するか否かを示し、変更を指示する場合は「true」が設定される。

【００３３】

図４の場合、「jobID」が「１」であるジョブで、１スロットを用いるMap処理タスク「1__m__1」が「Node１」のスレーブサーバ３０に割当てられる。そして、この「Node１」のスレーブサーバ３０は、「Node１」のスレーブサーバ３０と「Node２」のスレーブサーバ３０とからデータを取得して、Map処理を実行し、実行が完了していることを示す。

【００３４】

また、「jobID」が「１」であるジョブで、２番目に実行される１スロットを用いるReduce処理タスク「1__r__2」が「Node３」のスレーブサーバ３０に割当てられる。また、Reduce処理タスク「1__r__2」のデータ量は、「25000」であり、フラグに「true」が設定されている。そして、この「Node３」のスレーブサーバ３０は、Reduce処理タスクを分割して、並列に実行中であることを示す。

【００３５】

なお、JobID、TaskID、種別、Reduce項番については、ジョブリストＤＢ１２ａに記憶される情報にしたがって生成される。データのあるスレーブIDは、入出力ＤＢサーバ２が記憶するメタ情報等により特定することができる。状態は、タスクの割り当て状況やスレーブサーバ３０からの処理結果等によって更新される。割り当てスレーブIDは、タスクを割当時点で更新される。必要スロット数は、１タスクについて１スロットなどのように予め指定することができる。処理データ量は、Map処理の終了結果から予測することができる。フラグは、処理データ量が閾値を超えるか否かによって設定される。

【００３６】

制御部１３は、Map割当部１４、予測部１５、Reduce処理部１６を有する処理部であり、例えばプロセッサなどの電子回路である。また、制御部１３は、マスタサーバ１０全体の処理を司る。

【００３７】

Map割当部１４は、各ジョブにおけるMap処理のタスクであるMap処理タスクを１つ以上スレーブサーバ３０に割り当てる処理部である。具体的には、Map割当部１４は、データのあるスレーブＩＤの情報等を用いて、各Map処理タスクをスレーブサーバ３０に割り当てる。そして、Map割当部１４は、図４に示した「割当スレーブID」や「状態」等を更新する。

【００３８】

例えば、Map割当部１４は、スレーブサーバ３０等からMap処理タスクの割当要求を受信した場合に、タスクリストＤＢ１２ｂを参照して「状態」が「Not　assigned」のMap処理タスクを特定する。続いて、Map割当部１４は、割当要求を送信したスレーブサーバ３０のＩＤが「データのあるスレーブＩＤ」に含まれるMap処理タスクがあればそのMap処理タスクを優先して選び、そのようなMap処理タスクがなければ任意の方法でMap処理タスクを選び、割当対象のMap処理タスクとする。その後、Map割当部１４は、割当要求を送信したスレーブサーバ３０のIDを、割当対象のMap処理タスクの「スレーブサーバID」に格納する。

【００３９】

その後、Map割当部１４は、特定した割当先のスレーブサーバ３０に、TaskID、データのあるスレーブID、必要スロット数等を通知して、Map処理タスクを割当てる。また、Map割当部１４は、割当てたMap処理タスクの「状態」を「Not　assigned」から「Running」に更新する。

【００４０】

予測部１５は、Map処理タスクの実行結果を用いて、Reduce処理タスクの処理量を見積

もる処理部である。具体的には、予測部１５は、スレーブサーバ３０から通知されるMap処理の完了通知から、各Reduce処理タスクのデータ量を取得する。

【００４１】

予測部１５は、このようにして、所定数のMap処理の結果から取得されたReduce処理タスクのデータ量を加算して、Reduce処理タスクのデータ量を見積もる。そして、予測部１５は、見積もったReduce処理タスクのデータ量をタスクリストＤＢ１２ｂの処理データ量に格納し、Reduce処理タスクのデータ量が所定値以上であればフラグにtrueを格納する。なお、予測部１５は、完了通知を受信したMap処理タスクの「状態」を「Running」から「Done」に更新する。

【００４２】

図５は、Map処理の完了通知の例を示す図である。図５に示す完了通知は、各スレーブサーバ３０がマスタサーバ１０に送信する完了通知である。図５に示すように、完了通知は、「通知種別、JobID、完了Map　TaskID、Mapタスクを実行したスレーブID」から構成されるMap完了内容と、「通知種別、JobID、完了Map　TaskID、Reduce　TaskID、データ量」から構成されるMap完了内容とを含む。

【００４３】

ここで記憶される「通知種別」は、Map処理の完了通知かReduce情報かを示す情報であり、Map処理の完了通知の場合には「Map完了」が設定され、Reduce情報の場合には「Reduceデータ量」が設定される。「JobID」には、Map処理が属するジョブの識別子が設定される。「完了Map　TaskID」は、完了したMap処理タスクを特定する識別子が設定される。「Mapタスクを実行したスレーブID」は、当該Map処理タスクを実行し、完了通知を送信したスレーブサーバの識別子が設定される。「Reduce　TaskID」は、当該Map処理の実行結果からデータ量が判明したReduce処理タスクを特定する識別子が設定される。「データ量」は、当該Map処理の実行結果から判明したReduce処理タスクデータ量設定される。

【００４４】

図５の例は、「JobID」が「13」のジョブにおける「13__m__5」のMap処理タスクの完了結果を示している。このMap処理タスク「13__m__5」は、スレーブサーバ「Node１」で実行されたタスクである。また、このMap処理タスク「13__m__5」によって、「JobID」が「13」のジョブにおけるReduce処理が「13__r__1」、「13__r__2」、「13__r__3」の３つあることが判明したことを示す。さらに、このMap処理タスク「13__m__5」によって、Reduce処理タスク「13__r__1」のデータ量が「1000」、Reduce処理タスク「13__r__2」のデータ量が「1200」、Reduce処理タスク「13__r__3」のデータ量が「8000」であることが判明したことを示す。

【００４５】

予測部１５は、このようにしてMap処理タスクの完了通知からReduce処理タスクのデータ量を取得して加算していく。そして、予測部１５は、加算した結果が「10000」を超える場合に、フラグに「true」を設定する。

【００４６】

ここで、予測部１５がReduce処理の合計を見積もる契機を様々に設定することができる。つまり、どの程度のMap処理タスクが完了した時点で、処理データ量が閾値を超えるか否かを判定するのかを任意に設定することができる。

【００４７】

例えば、予測部１５は、全Map処理タスクのうち予め指定した割合のMap処理が完了した時点で判定することができる。また、予測部１５は、最初のMap処理タスクが終了してから予め指定した時間が過ぎて時点で判定することができる。また、予測部１５は、上記２つの時点のいずれか早い時点で判定することもできる。

【００４８】

なお、最初のMap処理タスクについても、ランダムに指定することもできる。このように、Map処理タスクのタスク数等に基づいて予測タイミングを任意に変更することができるので、入力データによってカスタマイズすることができる。

【００４９】
Reduce割当部１６は、スレーブサーバ３０からReduce処理タスクの割当要求を受信した場合に、Reduce処理タスクを割当てる処理部である。具体的にはReduce割当部１６は、振分キーに関するハッシュ関数等を用いて、各Reduce処理タスクをスレーブサーバ３０に割り当てる。そして、Reduce割当部１６は、図４に示した「割当スレーブID」や「状態」等を更新する。
【００５０】
例えば、Reduce割当部１６は、スレーブサーバ３０等からReduce処理タスクの割当要求を受信した場合に、タスクリストＤＢ１２ｂを参照して「状態」が「Not assigned」のReduce処理タスクを特定する。続いて、Reduce割当部１６は、ハッシュ関数等を用いて割当先のスレーブサーバを特定する。その後、Reduce割当部１６は、特定した割当先のスレーブサーバ３０のIDを、割当対象のReduce処理タスクの「スレーブサーバID」に格納する。
【００５１】
その後、Reduce割当部１６は、特定した割当先のスレーブサーバ３０に、TaskID、必要スロット数、処理データ量、フラグ等を通知して、Reduce処理タスクを割当てる。また、Reduce割当部１６は、割当てたMap処理タスクの「状態」を「Not assigned」から「Running」に更新する。なお、Reduce割当部１６は、Reduce処理タスクの完了通知を受信した場合、該当Reduce処理タスクの「状態」を「Running」から「Done」に更新する。
【００５２】
［スレーブサーバの構成］
図６は、実施例１に係るスレーブサーバの機能構成を示す機能ブロック図である。図６に示すように、スレーブサーバ３０は、通信制御部３１と、記憶部３２と、制御部３３とを有する。
【００５３】
通信制御部３１は、マスタサーバ１０や他のスレーブサーバ３０などと通信を実行する処理部であり、例えばネットワークインタフェースカードなどである。例えば、通信制御部３１は、マスタサーバ１０から各種タスクの割当を受信し、各種タスクの完了通知を送信する。また、通信制御部３１は、各種タスク処理の実行に伴って、該当するスレーブサーバ３０から読み出されたデータを受信する。
【００５４】
記憶部３２は、一時ファイルＤＢ３２ａと入出力ファイルＤＢ３２ｂとを有する記憶装置であり、例えばメモリやハードディスクなどである。また、記憶部３２は、制御部３３が実行するプログラムなどを記憶する。
【００５５】
一時ファイルＤＢ３２ａは、Map処理、Shuffle処理、Reduce処理等で生成される中間データ、他のスレーブサーバ３０等から読み出されたデータや各処理部が処理を実行する際に使用するデータを一時的に記憶するデータベースである。入出力ファイルＤＢ３２ｂは、Ｍａｐ処理の入力およびＲｅｄｕｃｅ処理の出力を記憶するデータベースであり、入出力ＤＢサーバ２と連携したデータベースである。
【００５６】
制御部３３は、Map処理部３４、Map結果送信部３５、Shuffle処理部３６、Reduce受信部３７、フラグ判定部３８、Reduce処理部３９を有する処理部であり、例えばプロセッサなどの電子回路である。また、制御部３３は、スレーブサーバ３０全体の処理を司る。
【００５７】
Map処理部３４は、Map処理タスクを実行する処理部である。具体的には、Map処理部３４は、ハートビートなどを用いて、マスタサーバ１０にMap処理タスクの割当を要求する。そして、Map処理部３４は、マスタサーバ１０から、「TaskID、データのあるスレーブID、必要スロット数」などを含むMap割当情報を受信する。
【００５８】

その後、Map処理部３４は、受信したMap割当情報にしたがって、「データのあるスレーブID」で特定されるスレーブサーバ３０が処理を行っているスレーブサーバであれば、入出力ＤＢ３２ｂからデータを取得し、そうでなければ、「データのあるスレーブＩＤ」で特定されるスレーブサーバ３０からデータを取得して一時ファイルＤＢ３２ａ等に保存し、「必要スロット数」で指定されるスロット数を用いてMap処理を実行する。そして、Map処理部３４は、Map処理結果を一時ファイルＤＢ３２ａ等に格納する。ここで、生成されるMap処理結果は、例えば図５に示すように、Reduce処理のタスクIDやデータ量などが含まれる。

【００５９】

Map結果送信部３５は、Map処理部３４が実行したMap処理の結果をマスタサーバ１０に送信する処理部である。例えば、Map結果送信部３５は、Map処理部３４からMap処理が終了したことが通知されると、一時ファイルＤＢ３２ａなどからMap処理結果の一部を読み出す。そして、Map結果送信部３５は、図５に示した完了通知を生成してマスタサーバ１０に送信する。

【００６０】

Shuffle処理部３６は、Map処理の結果を「Key」でソートし、同じ「Key」を有する「Key、Value」ペアをマージして、Reduce処理の処理対象を生成する処理部である。具体的には、Shuffle処理部３６は、マスタサーバ１０からMap処理が終了したことを通知されると、当該Map処理が属するジョブのReduce処理を実行する準備として、各スレーブサーバ３０から該当するMap処理結果を取得する。そして、Shuffle処理部３６は、Map処理の結果を予め指定された「Key」でソートし、同じ「Key」を有する処理結果をマージして、一時ファイルＤＢ３２ａに格納する。

【００６１】

例えば、Shuffle処理部３６は、「JobID」が「１」のMap処理タスクである「1__m__1、1__m__2、1__m__3、1__m__4」が終了したこと、つまり、「JobID」が「１」のReduce処理タスクの実行開始をマスタサーバ１０から受信する。すると、Shuffle処理部３６は、Node1、Node2、Node3、Node4からMap処理結果を取得する。続いて、Shuffle処理部３６は、Map処理結果のソートおよびマージを実行し、その結果を一時ファイルＤＢ３２ａ等に格納する。

【００６２】

Reduce受信部３７は、マスタサーバ１０から割当てられたReduce処理タスクを受信する処理部である。例えば、Reduce受信部３７は、「JobID、TaskID、必要スロット数、処理データ量、フラグ」などから構成されるReduce処理タスクの情報を受信する。そして、Reduce受信部３７は、受信した情報を一時ファイルＤＢ３２ａ等に格納する。

【００６３】

フラグ判定部３８は、マスタサーバ１０から割当てられたReduce処理タスクにフラグが設定されているか否かを判定する処理部である。具体的には、フラグ判定部３８は、Reduce受信部３７が一時ファイルＤＢ３２ａ等に格納したReduce処理タスクの情報を参照して、フラグが設定されているか否かを判定する。そして、フラグ判定部３８は、判定結果をReduce処理部３９に通知する。

【００６４】

例えば、フラグ判定部３８は、Reduce処理タスクの情報が「JobID＝2、TaskID＝2__r__1、必要スロット数＝1、処理データ量＝24000、フラグ＝true」であった場合、「フラグ＝true」であることから、フラグが設定されていると判定する。なお、フラグ判定部３８は、「フラグ」に「true」が設定されていない場合、フラグが設定されていないと判定する。

【００６５】

Reduce処理部３９は、フラグ判定部３８による判定結果に基づいて、Reduce処理タスクの処理方法を変更して、Reduce処理タスクを実行する処理部である。具体的には、Reduce処理部３９は、割当てられたReduce処理タスクにフラグが設定されている場合には、割当

てられたReduce処理タスクを分散処理する。一方、Reduce処理部３９は、割当てられたReduce処理タスクにフラグが設定されていない場合には、割当てられたReduce処理タスクを分散処理せずに実行する。そして、Reduce処理部３９は、Reduce処理タスクの処理結果を入出力ファイルＤＢ３２ｂ等に格納する。

【００６６】

例えば、Reduce処理部３９は、スレーブサーバ３０が有するプロセッサの数、ディスクの数、予め指定された数の少なくとも１つを用いて、割り当てられたReduce処理タスクをサブタスクに分割して、複数のプロセッサで並列実行する。

【００６７】

一例を挙げると、Reduce処理部３９は、プロセッサの数またはディスクの数が４つである場合、Reduce処理タスクを４つのサブタスクに分割し、４つプロセッサを用いて各サブタスクを並列に実行する。

【００６８】

（Map処理の説明）

ここで、スレーブサーバ３０が実行するMap処理について説明する。図７は、Map処理を説明する図である。図７に示すように、各スレーブサーバ３０は、入力データとして「Hello Apple！」と「Apple is red」を受信し、それぞれの入力データに対してMap処理を実行して、「Key、Value」のペアを出力する。

【００６９】

図７の例では、スレーブサーバ３０は、「Hello Apple！」に対してMap処理を実行して、入力データの各要素の数を計数し、要素を「Key」、計数結果を「Value」とする「Key、Value」のペアを出力する。具体的には、スレーブサーバ３０は、入力データ「Hello Apple！」から「Hello、1」、「Apple、1」、「！、1」を生成する。同様に、スレーブサーバ３０は、入力データ「Apple is red」から「Apple、1」、「is、1」、「red、1」を生成する。

【００７０】

（Shuffle処理）

次に、スレーブサーバ３０が実行するShuffle処理について説明する。図８は、Shuffle処理を説明する図である。図８に示すように、各スレーブサーバ３０は、各スレーブサーバからMap処理結果を取得してShuffle処理を実行する。

【００７１】

図８の例では、スレーブサーバ（Ａ）、（Ｂ）、（Ｃ）・・・が同じジョブ（例えば、JobID＝20）に属するMap処理タスクを実行し、スレーブサーバ（Ｄ）と（Ｚ）とが、JobID＝20に属するReduce処理タスクを実行する。

【００７２】

例えば、スレーブサーバ（Ａ）がMap処理１を実行して「Apple、1」、「is、3」を生成し、スレーブサーバ（Ｂ）がMap処理２を実行して「Apple、2」、「Hello、4」を生成し、スレーブサーバ（Ｃ）がMap処理３を実行して「Hello、3」、「red、5」を生成する。スレーブサーバ（Ｘ）がMap処理１０００を実行して「Hello、1000」、「is、1002」を生成する。

【００７３】

続いて、スレーブサーバ（Ｄ）およびスレーブサーバ（Ｚ）は、割当てられたReduce処理タスクで使用する各スレーブサーバのMap処理結果を取得して、ソートおよびマージを実行する。具体的には、スレーブサーバ（Ｄ）には、「Apple」と「Hello」についてのReduce処理タスクが割当てられて、スレーブサーバ（Ｚ）には、「is」と「red」についてのReduce処理タスクが割当てられたとする。

【００７４】

この場合、スレーブサーバ（Ｄ）は、スレーブサーバ（Ａ）からMap処理１の結果「Apple、1」を取得し、スレーブサーバ（Ｂ）からMap処理２の結果「Apple、2」および「Hello、4」を取得する。また、スレーブサーバ（Ｄ）は、スレーブサーバ（Ｃ）からMap処理

３の結果「Hello、3」を取得し、スレーブサーバ（Ｘ）からMap処理１０００の結果「Hello、1000」を取得する。そして、スレーブサーバ（Ｄ）は、これらの結果をソートおよびマージして、「Apple、［1,2］」および「Hello、［3,4,1000］」を生成する。

【００７５】

同様に、スレーブサーバ（Ｚ）は、スレーブサーバ（Ａ）からMap処理１の結果「is、3」を取得し、スレーブサーバ（Ｃ）からMap処理３の結果「red、5」を取得し、スレーブサーバ（Ｘ）からMap処理１０００の結果「is、1002」を取得する。そして、スレーブサーバ（Ｚ）は、これらの結果をソートおよびマージして、「is、［3,1002］」および「red、［5］」を生成する。

【００７６】

（Reduce処理）

次に、スレーブサーバ３０が実行するReduce処理について説明する。図９は、Reduce処理を説明する図である。図９に示すように、各スレーブサーバ３０は、各スレーブサーバのMap処理結果から生成したShuffle結果を用いて、Reduce処理を実行する。具体的には、Shuffle処理の説明と同様、スレーブサーバ（Ｄ）には、「Apple」と「Hello」についてのReduce処理タスクが割当てられて、スレーブサーバ（Ｚ）には、「is」と「red」についてのReduce処理タスクが割当てられたとする。

【００７７】

この例では、スレーブサーバ（Ｄ）は、Shuffle処理の結果である「Apple、［1,2］」および「Hello、［3,4,1000］」から、Reduce処理結果として「Apple、3」および「Hello、1007」を生成する。同様に、スレーブサーバ（Ｚ）は、Shuffle処理の結果である「is、［3,1002］」および「red、［5］」から、Reduce処理結果として「is、1005」および「red、5」を生成する。

【００７８】

（Reduce処理タスクのフラグ設定）

次に、マスタサーバ１０がMap処理結果からReduce処理タスクにフラグを設定する例を説明する。図１０は、Reduce処理タスクの処理量を予測してフラグを設定する処理を説明する図である。この図１０は、マスタサーバ１０が保持するタスクリストを示している。

【００７９】

図１０示すタスクリストのうち「JobID＝１」についてはReduce処理の割当が完了しており、Reduce処理が既に実行されていることを示す。このような状態で、「JobID＝２」におけるMap処理タスク「2__m__1、2__m__2、2__m__3、2__m__4」のうち「2__m__1」および「2__m__2」が完了したとする（Ｓ１）。

【００８０】

そして、マスタサーバ１０は、Map処理タスク「2__m__1」を実行したNode1からReduce処理のデータ量を含むMap完了通知１を受信し、Map処理タスク「2__m__2」を実行したNode2からReduce処理のデータ量を含むMap完了通知２を受信する（Ｓ２）。

【００８１】

続いて、マスタサーバ１０は、受信したMap完了通知１とMap完了通知２とから、「JobID＝２」におけるReduce処理タスク「2__r__1」と「2__r__2」の処理データ量をそれぞれ「24000」と「13000」と見積もる（Ｓ３）。

【００８２】

その後、マスタサーバ１０は、処理データ量が閾値「20000」を超えるReduce処理タスク「2__r__1」に対して、フラグ「true」を設定する（Ｓ４）。そして、マスタサーバ１０は、Reduce処理タスク「2__r__1」と「2__r__2」の割当先をハッシュ関数で決定し、Reduce処理タスク「2__r__1」が割当てられたNodeに、Reduce処理タスクとともにフラグ「true」を送信する（Ｓ５）。

【００８３】

このように、マスタサーバ１０は、ジョブにおける全てのMap処理タスクが終了する前に、一部のMap処理タスクが完了した時点で、Reduce処理タスクの処理データ量を見積り

、見積もった結果に応じてフラグを設定する。
【００８４】
［マスタサーバの処理］
図１１は、実施例１に係るマスタサーバが実行する処理の流れを示すフローチャートである。図１１に示すように、マスタサーバ１０は、管理者等から登録されるジョブ登録の情報にしたがって、ジョブリストＤＢ１２ａにジョブリストやタスクリストＤＢ１２ｂにタスクリストを追加する（Ｓ１０１）。
【００８５】
その後、マスタサーバ１０は、スレーブサーバ３０からハートビートなどの通知を受信するまで待機し（Ｓ１０２）、通知を受信した場合に、当該通知がタスク要求かタスクの完了通知かを判定する（Ｓ１０３）。
【００８６】
そして、マスタサーバ１０のMap割当部１４は、受信された通知がタスク要求であると判定された場合（Ｓ１０３：タスク要求）、ハッシュ関数等を用いてタスク割当を実施する（Ｓ１０４）。その後、Map割当部１４は、通知要求元のスレーブサーバ３０に、割当てたタスク情報を応答する（Ｓ１０５）。ここで、タスク情報には、タスクに属するジョブに関するジョブリストの該当行の一行分の情報およびタスクリストの該当行の一行分の情報などが含まれる。
【００８７】
一方、マスタサーバ１０の制御部１３は、受信された通知がタスクの完了通知であると判定された場合（Ｓ１０３：完了通知）、該当タスクの完了処理を実行する（Ｓ１０６）。その後、制御部１３は、ジョブの全タスクが完了した場合（Ｓ１０７：Ｙｅｓ）、Ｓ１０１に戻って以降の処理を繰り返す。一方、制御部１３は、ジョブの全タスクが完了していない場合（Ｓ１０７：Ｎｏ）、Ｓ１０２に戻って以降の処理を繰り返す。
【００８８】
（タスクの完了処理）
図１２は、マスタサーバが実行する該当タスクの完了処理の流れを示すフローチャートである。この処理は、図１１のＳ１０６で実行される処理である。
【００８９】
図１２に示すように、マスタサーバ１０の予測部１５は、受信された完了通知がMap処理タスクの完了通知であると判定された場合（Ｓ２０１：Map）、終了したMap処理の完了通知からReduce処理タスクの処理データ量を加算する（Ｓ２０２）。
【００９０】
例えば、予測部１５は、受信された完了通知のヘッダにMap処理タスクを示す識別子が付与されていた場合に、Map処理タスクの完了通知を判定し、当該完了通知に含まれるReduceデータ量に基づいて、図４に示す該当Reduce処理のデータ量を計算する。
【００９１】
その後、予測部１５は、所定のMap処理タスクが完了し、フラグ判定タイミングであると判定すると（Ｓ２０３：Ｙｅｓ）、図４を参照し、処理データ量が所定値を超えるReduce処理タスクが存在するか否かを判定する（Ｓ２０４）。
【００９２】
そして、予測部１５は、処理（転送）データ量が所定値を超えるReduce処理タスクにフラグ「true」を設定して、フラグの付いたReduce処理タスクの割当を実行する（Ｓ２０５）。例えば、予測部１５は、ハッシュ値等を用いて、フラグ「true」を設定したReduce処理タスクの割当先を決定し、決定した割当先のスレーブサーバ３０にReduce処理タスクを送信する。
【００９３】
続いて、予測部１５は、完了通知を受信したMap処理タスクの「状態」を「Done」に変更し（Ｓ２０６）、該当Map処理タスクの完了をMap処理タスクの完了通知領域に登録する（Ｓ２０７）。

【００９４】
一方、マスタサーバ１０のReduce割当部１６は、受信された完了通知がMap処理タスクではなく、Reduce処理タスクの完了通知であると判定された場合（Ｓ２０１：Reduce）、完了通知を受信したReduce処理タスクの「状態」を「Done」に変更する（Ｓ２０８）。
【００９５】
［スレーブサーバの処理］
図１３は、実施例１に係るスレーブサーバが実行する処理の流れを示すフローチャートである。図１３に示すように、スレーブサーバ３０は、マスタサーバ１０に、ハートビートでタスク要求を送信する（Ｓ３０１）。
【００９６】
続いて、スレーブサーバ３０は、タスク要求の応答としてジョブ情報とタスク情報を取得し（Ｓ３０２）、取得したタスク情報がMap処理タスクの情報か否かを判定する（Ｓ３０３）。
【００９７】
そして、スレーブサーバ３０のMap処理部３４は、取得したタスク情報がMap処理タスクの情報であると判定された場合（Ｓ３０３：Map）、入力データを読み込み（Ｓ３０４）、Map処理タスクを起動する（Ｓ３０５）。
【００９８】
例えば、Map処理部３４は、取得したMap処理タスクの情報における「データのあるスレーブID」で特定されるスレーブサーバ３０から入力データを取得し、取得したMap処理タスクの情報によって割当てられたMap処理タスクを起動する。
【００９９】
その後、Map処理部３４は、一時ファイルＤＢ３２ａに、Reduce処理タスクごとに分けて処理結果を保存し（Ｓ３０６）、Map処理タスクが終了するまで待機する（Ｓ３０７）。そして、Map結果送信部３５は、ハートビート等でMap処理タスクの完了とReduce向けデータ量をマスタサーバ１０に送信する（Ｓ３０８）。
【０１００】
一方、スレーブサーバ３０のReduce受信部３７が取得したタスク情報がReduce処理タスクの情報であると判定した場合（Ｓ３０３：Reduce）、Shuffle処理部３６が、各スレーブサーバ３０からMap処理結果を取得して、Shuffle処理を実行する（Ｓ３０９）。
【０１０１】
Reduce処理部３９は、Reduce受信部３７が取得したReduce処理タスクを実行して（Ｓ３１０）、タスクが終了するまで待機し（Ｓ３１１）、タスクが完了すると、ハートビート等で完了通知をマスタサーバ１０に送信する（Ｓ３１２）。
【０１０２】
（Reduce処理タスクの起動処理）
図１４は、スレーブサーバが実行するReduce処理タスクの起動処理の流れを示すフローチャートである。図１４に示すように、フラグ判定部３８は、Reduce受信部３７が受信したReduce処理タスクにフラグ「true」が付加されているか否かを判定する（Ｓ４０１）。
【０１０３】
そして、フラグ判定部３８がReduce処理タスクにフラグ「true」が付加されていると判定した場合（Ｓ４０１：Ｙｅｓ）、Reduce処理部３９は、Reduce処理タスクの入力を分割する（Ｓ４０２）。
【０１０４】
続いて、Reduce処理部３９は、各分割された入力でＳ４０３からＳ４０５までループ処理する。具体的には、Reduce処理部３９は、各分割されたReduce処理タスクの入力に関して、Reduce処理タスクのサブタスクを起動する（Ｓ４０４）。
【０１０５】
そして、Reduce処理部３９は、Reduce処理タスクの全サブタスクが完了するまで待機し（Ｓ４０６）、全サブタスクが完了すると、処理を終了する。

【０１０６】
一方、Ｓ４０１において、フラグ判定部３８がReduce処理タスクにフラグ「true」が付加されていないと判定した場合（Ｓ４０１：Ｎｏ）、Reduce処理部３９は、受信されたReduce処理タスクをそのまま実行する（Ｓ４０７）。そして、Reduce処理部３９は、Reduce処理タスクが完了すると、処理を終了する。
【０１０７】
（Reduce処理タスクの分割処理）
図１５は、スレーブサーバが実行するReduce処理タスクの分割処理の流れを示すフローチャートである。図１５に示すように、スレーブサーバ３０のReduce処理部３９は、自サーバ内におけるスレーブの受け入れ可能スロット数を「Ｓ」と設定する（Ｓ５０１）。
【０１０８】
そして、Reduce処理部３９は、変数「ｉ」がＳ－１となるまで順にＳ５０２からＳ５０８までを実行するループ処理する。具体的には、Reduce処理部３９は、「ｉ×Reduce処理タスクの入力の全レコード数／Ｓ」を「開始位置」に設定する（Ｓ５０３）。その後、Reduce処理部３９は、Reduce処理タスクの入力「開始位置」が「Key」ではない間、Ｓ５０４からＳ５０６の処理を繰り返すループ処理を実行する。
【０１０９】
具体的には、Reduce処理部３９は、Reduce処理タスクの入力「開始位置」が「Key」になるまで、「開始位置」をインクリメントする（Ｓ５０５）。そして、Reduce処理部３９は、Reduce処理タスクの入力「開始位置」が「Key」になると、分割された入力の開始位置「ｉ」に、Ｓ５０４からＳ５０６で算出された「開始位置」を代入する（Ｓ５０７）。その後、Reduce処理部３９は、Ｓ５０２以降のループ処理を実行する。
【０１１０】
上述したように、マスタサーバ１０は、一部のMap処理タスクの実行後に、データ量の集中する可能性が高いReduce処理タスクを検知する。そして、マスタサーバ１０は、検知したReduce処理タスクをスレーブサーバ３０で処理させる際に、スレーブサーバ３０内で並列実行させることができる。
【０１１１】
このように、マスタサーバ１０は、タスク割り当ての際には知ることができないReduceタスクの処理量の不均一性に関する情報を、各スレーブサーバ３０に伝えて処理方法を変更させることができる。したがって、該当スレーブサーバのみReduce処理タスクを優先させることにより、ジョブ全体の完了時間を短縮することができる。
【０１１２】
また、マスタサーバ１０は、全Map処理タスクの所定の割合が終了した場合や最初のMap処理タスクが終了してから所定時間が経過した場合に、Reduce処理タスクの処理量を見積もることができる。この結果、マスタサーバ１０は、どの程度のMap処理タスクが終了したところで、フラグ付けの判断を行うかを任意の方式で決定することができるので、ジョブに適した方式を採用することができ、汎用性を高めることができる。
【０１１３】
スレーブサーバ３０は、処理量の多いReduce処理タスクについては、フラグが通知されるので、当該Reduce処理タスクを分割して実行することができ、処理時間を短縮することができる。また、スレーブサーバ３０は、Reduce処理タスクを分割する際に、プロセッサの数、ディスクの数、予め指定された数の少なくとも１つを用いて分割するので、自サーバの処理性能にあわせて分割することができる。
【実施例２】
【０１１４】
ところで、マスタサーバ１０は、フラグ付きのReduce処理タスクを割当てる先のスレーブサーバ３０に、他のReduce処理タスクが既に割当てられている場合に、フラグ付きのReduce処理タスクを優先させることができる。
【０１１５】

そこで、実施例２では、フラグ付きのReduce処理タスクを優先させる例について説明する。図１６は、実施例２に係るReduce処理タスクの割当処理の流れを示すフローチャートである。

【０１１６】

図１６に示すように、マスタサーバ１０のReduce割当部１６は、タスクリストＤＢ１２ｂ中のReduce処理タスクを処理データ量が多い順に並び替える（Ｓ６０１）。続いて、Reduce割当部１６は、タスクリストＤＢ１２ｂ中のReduce処理タスクを処理データ量の多い順に、スレーブサーバ３０の数だけ選択し、「優先タスクリストＰ」と設定する（Ｓ６０２）。そして、Reduce割当部１６は、「未割当優先タスク数」に「０」を設定する（Ｓ６０３）。

【０１１７】

その後、Reduce割当部１６は、「優先タスクリストＰ」の数のReduce処理タスクについて、Ｓ６０４からＳ６０８までをループ処理する。具体的には、Reduce割当部１６は、処理対象のReduce処理タスクが既に割当済みである場合（Ｓ６０５：Ｙｅｓ）、当該タスクと同じスレーブサーバ３０に割当てられた他のReduce処理タスクを中止する（Ｓ６０６）。一方、Reduce割当部１６は、処理対象のReduce処理タスクが割当済みではない場合（Ｓ６０５：Ｎｏ）、「未割当優先タスク数」をインクリメントする（Ｓ６０７）。

【０１１８】

Ｓ６０５からＳ６０８までのループ処理が終了すると、Reduce割当部１６は、「未割当優先タスク数」が０より大きいか否かを判定する（Ｓ６０９）。そして、Reduce割当部１６は、「未割当優先タスク数」が０である場合（Ｓ６０９：Ｎｏ）、処理を終了する。

【０１１９】

一方、Reduce割当部１６は、「未割当優先タスク数」が０より大きい場合（Ｓ６０９：Ｙｅｓ）、全スレーブサーバ３０に対してＳ６１０からＳ６１５までをループ処理する。

【０１２０】

具体的には、Reduce割当部１６は、該当スレーブサーバ３０に優先タスクリストのいずれかのReduce処理タスクが割当済みである場合には（Ｓ６１１：Ｙｅｓ）、当該スレーブサーバ３０に対するループ処理を終了し（Ｓ６１５）、次のスレーブサーバ３０に対してループ処理を実行する。

【０１２１】

一方、Reduce割当部１６は、該当スレーブサーバ３０に、優先タスクリストにおけるいずれのReduce処理タスクも割当てられていない場合には（Ｓ６１１：Ｎｏ）、対象のスレーブサーバ３０に割当てられた他のReduce処理タスクを中止する（Ｓ６１２）。続いて、Reduce割当部１６は、該当する優先タスクを対象のスレーブサーバ３０に割り与え、「未割当優先タスク数」を１減算する（Ｓ６１３）。

【０１２２】

その後、Reduce割当部１６は、１減算後の「未割当優先タスク数」が０である場合には（Ｓ６１４：Ｙｅｓ）、処理を終了する。一方、Reduce割当部１６は、１減算後の「未割当優先タスク数」が０より大きい場合には（Ｓ６１４：Ｎｏ）、次のスレーブサーバ３０についてＳ６１０以降のループ処理を実行する。

【０１２３】

このように、マスタサーバ１０は、フラグ付きのReduce処理タスクを割当てる先のスレーブサーバ３０に、他のReduce処理タスクが既に割当てられている場合に、フラグ付きのReduce処理タスクを優先させることができる。このため、マスタサーバ１０は、後からフラグが付いたReduce処理タスクを優先させて処理させることができるので、Reduce処理タスクの見積り順序に依存せずに、ジョブ全体の完了時間を短縮することができる。

【０１２４】

一般には実行中のタスクを中止することで無駄になる計算時間、また送信済みのデータを使わなくなるために無駄になる通信時間やデータの読み書き時間が発生する。しかし、いくつかのMap処理タスクが終了してフラグが立った段階で、フラグの経ったReduce処理

タスクと同じスレーブサーバ３０に割り当て済みのReduce処理タスクを中止するようにすることで、一部のデータ転送の無駄とそのデータの処理の無駄だけに抑えることができる。また、クリティカルパス上にあってもっとも早く開始したいフラグの立ったReduce処理タスクの開始を従来通り素早く開始できるため全体としての処理時間が短くなる。

【実施例３】

【０１２５】

ところで、マスタサーバ１０は、Reduce処理タスクのフラグが設定されてすぐに割当を実施せずに、いくつかのMap処理タスクが終了して割当てることもできる。そこで、実施例３では、すぐにReduce処理タスクの割当を実行せずに、いくつかのMap処理タスクが終了してからReduce処理タスクの割当を実行する例を説明する。

【０１２６】

図１７は、実施例３に係るReduce処理タスクの割当処理の流れを示すフローチャートである。図１７に示すように、マスタサーバ１０のMap割当部１４は、タスク要求してきたスレーブサーバ３０がMap処理タスクの受け入れが可能である場合（Ｓ７０１：Ｙｅｓ）、未割り当てのローカルMap処理タスクが存在するか否かを判定する（Ｓ７０２）。ここで、ローカルMap処理タスクとは、タスク要求してきたスレーブサーバ３０がタスクリスト中の「データがあるスレーブID」の列に含まれているタスクを指す。

【０１２７】

そして、Map割当部１４は、未割り当てのローカルMap処理タスクが存在すると判定した場合（Ｓ７０２：Ｙｅｓ）、タスク要求してきたスレーブサーバ３０に、当該未割り当てのローカルMap処理タスクを割当てる（Ｓ７０３）。その後、Map割当部１４は、Ｓ７０１以降の処理を繰り返す。

【０１２８】

一方、Map割当部１４は、未割り当てのローカルMap処理タスクが存在せず（Ｓ７０２：Ｎｏ）、未割り当てのMap処理タスクが存在すると判定した場合（Ｓ７０４：Ｙｅｓ）、タスク要求してきたスレーブサーバに、未割り当てのMap処理タスクを割当てる（Ｓ７０３）。

【０１２９】

そして、Map処理タスクの受け入れが可能ではなく（Ｓ７０１：Ｎｏ）、または、未割り当てのMap処理タスクが存在しない場合（Ｓ７０４：Ｎｏ）、Ｓ７０５を実行する。具体的には、Reduce割当部１６は、タスク要求してきたスレーブサーバ３０がReduce処理タスクの受け入れが可能であるか否かを判定する。ここで、Reduce処理タスクの受け入れが可能であるとは、「スレーブサーバ３０の処理可能なReduce処理タスク数＞スレーブサーバ３０に割り当て済みReduce処理タスクの必要スロット数の合計」である状態を指す。

【０１３０】

Reduce割当部１６は、タスク要求してきたスレーブサーバ３０がReduce処理タスクの受け入れが可能であると判定した場合（Ｓ７０５：Ｙｅｓ）、Ｓ７０６を実行する。具体的には、Reduce割当部１６は、ジョブリストＤＢ１２ａのReduce割当許可がtrueか否かを参照して、Reduce処理タスクの割当が許可されているか否かを判定する。

【０１３１】

そして、Reduce割当部１６は、Reduce処理タスクの割当が許可されていると判定した場合（Ｓ７０６：Ｙｅｓ）、未割り当てのReduce処理タスクが存在するか否かを判定する（Ｓ７０７）。

【０１３２】

ここで、Reduce割当部１６は、未割り当てのReduce処理タスクが存在すると判定した場合（Ｓ７０７：Ｙｅｓ）、タスク要求してきたスレーブサーバ３０に、未割り当てのReduce処理タスクを割り当てる（Ｓ７０８）。その後、Reduce割当部１６は、Ｓ７０１以降の処理を繰り返す。

【０１３３】

一方、Reduce処理タスクの受け入れが可能ではない場合（Ｓ７０５：Ｎｏ）、Reduce処

理タスクの割当が許可されていない場合（Ｓ７０６：Ｎｏ）、未割り当てのReduce処理タスクが存在しない場合（Ｓ７０７：Ｎｏ）、Reduce割当部１６は、処理を終了する。

【０１３４】

このように、マスタサーバ１０は、Reduce処理タスクのフラグが設定されてすぐに割当を実施せずに、いくつかのMap処理タスクが終了して割当てることもできる。この方式では、shuffle処理のデータ転送の開始が遅くなる場合もあるが、多くのMap処理タスクのデータ量は均等であり、最初に割り当てられたMapタスクは一斉に終わることが期待される。このため、遅れの影響は限定的である一方、中止されるReduce処理タスクはないので、データ転送とその処理に関する無駄は発生しない。

【０１３５】

したがって、若干の余分な処理時間がかかる場合があるが、ボトルネックとなるデータ量の多いReduce処理タスクの処理時間を短縮することで、ジョブ全体の処理時間を短縮できる。

【実施例４】

【０１３６】

さて、これまで本発明の実施例について説明したが、本発明は上述した実施例以外にも、種々の異なる形態にて実施されてよいものである。そこで、以下に異なる実施例を説明する。

【０１３７】

（Reduce処理タスクのフラグ判断）

マスタサーバ１０は、所定値以上の処理データ量であるReduce処理タスクをフラグ対象として決定する例を説明したが、これに限定されず、様々な判断基準で決定することができる。

【０１３８】

例えば、マスタサーバ１０は、Reduce処理タスクのデータ量の平均を「ｍ」とした場合、例えば２などのあらかじめ定められた係数ｋ基づき、「ｋ×ｍ」よりも多くのデータ量を有するReduce処理タスクにフラグを付けることもできる。

【０１３９】

また、マスタサーバ１０は、Reduce処理タスクのデータ量の平均を「ｍ」、分散を「σ」とした場合に、例えば３などのあらかじめ定められた係数ｋに基づき、「ｍ＋ｋ×σ」よりも多くのデータ量を有するReduce処理タスクにフラグを付けることもできる。

【０１４０】

また、マスタサーバ１０は、スレーブタスク３０の同時実行可能タスク数を「ｓ」、最大のReduce処理タスクのデータ量を「ｄ」とした場合に、サブタスクのデータ量「ｄ／ｓ＝ｄ´」とすると、「ｄ´」より多くのデータ量を持つ全てのReduce処理タスクにフラグを付けることもできる。

【０１４１】

（システム）

また、本実施例において説明した各処理のうち、自動的におこなわれるものとして説明した処理の全部または一部を手動的におこなうこともできる。あるいは、手動的におこなわれるものとして説明した処理の全部または一部を公知の方法で自動的におこなうこともできる。この他、上記文書中や図面中で示した処理手順、制御手順、具体的名称、各種のデータやパラメータを含む情報については、特記する場合を除いて任意に変更することができる。

【０１４２】

また、図示した各装置の各構成要素は機能概念的なものであり、必ずしも物理的に図示の如く構成されていることを要しない。すなわち、各装置の分散や統合の具体的形態は図示のものに限られない。つまり、その全部または一部を、各種の負荷や使用状況などに応じて、任意の単位で機能的または物理的に分散・統合して構成することができる。さらに、各装置にて行なわれる各処理機能は、その全部または任意の一部が、ＣＰＵおよび当該

ＣＰＵにて解析実行されるプログラムにて実現され、あるいは、ワイヤードロジックによるハードウェアとして実現され得る。

【０１４３】

（ハードウェア）

次に、各サーバのハードウェア構成例を説明するが、各サーバは同様の構成を有するので、ここでは一例を説明する。図１８は、サーバのハードウェア構成例を示す図である。図１７に示すように、サーバ１００は、通信インタフェース１０１、メモリ１０２、複数のＨＤＤ（ハードディスクドライブ）１０３、プロセッサ装置１０４を有する。

【０１４４】

通信インタフェース１０１は、図２や図６に示した通信制御部に該当し、例えばネットワークインタフェースカードなどである。複数のＨＤＤ１０３は、図２や図６に示した機能を動作させるプログラムやＤＢ等を記憶する。

【０１４５】

プロセッサ装置１０４が有する複数のＣＰＵ１０５は、図２や図６に示した各処理部と同様の処理を実行するプログラムをＨＤＤ１０３等から読み出してメモリ１０２に展開することで、図２や図６等で説明した各機能を実行するプロセスを動作させる。すなわち、このプロセスは、マスタサーバ１０が有するMap割当部１４、予測部１５、Reduce処理部１６と同様の機能を実行する。また、このプロセスは、スレーブサーバ３０が有するMap処理部３４、Map結果送信部３５、Shuffle処理部３６、Reduce処理部３７、フラグ判定部３８、Reduce処理部３９と同様の機能を実行する。

【０１４６】

このようにサーバ１００は、プログラムを読み出して実行することで、タスク割当方法またはタスク実行方法を実行する情報処理装置として動作する。また、サーバ１００は、媒体読取装置によって記録媒体から上記プログラムを読み出し、読み出された上記プログラムを実行することで上記した実施例と同様の機能を実現することもできる。なお、この他の実施例でいうプログラムは、サーバ１００によって実行されることに限定されるものではない。例えば、他のコンピュータまたはサーバがプログラムを実行する場合や、これらが協働してプログラムを実行するような場合にも、本発明を同様に適用することができる。

【符号の説明】

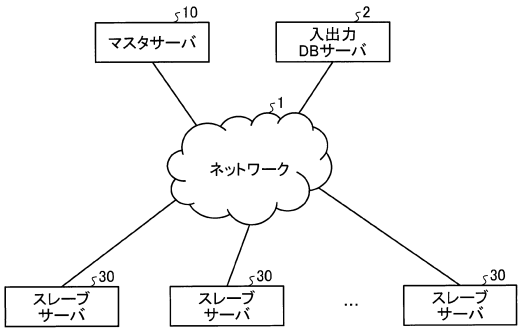
【０１４７】

１０　マスタサーバ
１１　通信制御部
１２　記憶部
１２ａ　ジョブリストＤＢ
１２ｂ　タスクリストＤＢ
１３　制御部
１４　Map割当部
１５　予測部
１６　Reduce割当部
３０　スレーブサーバ
３１　通信制御部
３２　記憶部
３２ａ　一時ファイルＤＢ
３２ｂ　入出力ファイルＤＢ
３３　制御部
３４　Map処理部
３５　Map結果送信部
３６　Shuffle処理部
３７　Reduce受信部

３８　フラグ判定部
３９　Reduce処理部

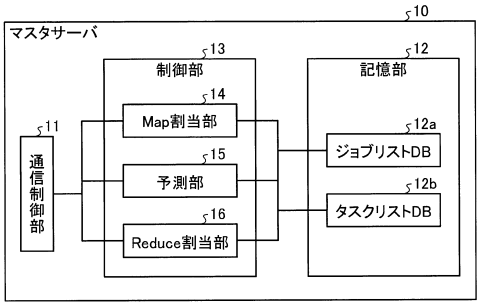
【図１】

実施例1に係る分散処理システムの全体構成例を示す図



【図２】

実施例1に係るマスタサーバの機能構成を示す機能ブロック図



【図３】

ジョブリストDBに記憶される情報の例を示す図

JobID	総Mapタスク数	総Reduce タスク数	Reduce 割当許可
1	4	2	false
2	4	2	false

【図４】

タスクリストDBに記憶される情報の例を示す図

JobID	TaskID	種別	Reduceの項番	データのあるスレーブID	状態	割り当てスレーブID	必要スロット数	処理データ量	フラグ
1	1_m_1	Map	–	Node1, Node2	Done	Node1	1		
1	1_m_2	Map	–	Node3, Node4	Done	Node4	1		
1	1_m_3	Map	–	Node1, Node3	Running	Node1	1		
1	1_m_4	Map	–	Node2, Node4	Running	Node2	1	12000	
1	1_r_1	Reduce	1	–	Running	Node2	1	25000	true
1	1_r_2	Reduce	2	–	Running	Node3	1		
2	2_m_1	Map	–	Node3, Node4	Done	Node1	1		
2	2_m_2	Map	–	Node1, Node3	Done	Node2	1		
2	2_m_3	Map	–	Node2, Node4	Running	Node3	1		
2	2_m_4	Map	–	Node1, Node4	Running	Node4	1		
2	2_r_1	Reduce	1	–	Not assigned	–	1	24000	true
2	2_r_2	Reduce	2	–	Not assigned	–	1	13000	

【図５】

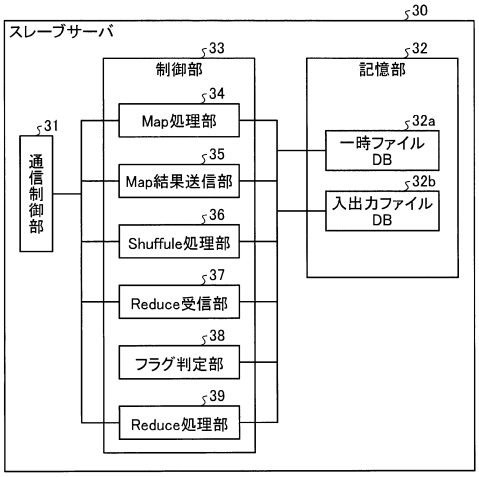
Map処理の完了通知の例を示す図

通知種別	JobID	完了Map TaskID	Mapタスクを実行したスレーブID
Map完了	13	13_m_5	Node1

通知種別	JobID	完了Map TaskID	Reduce TaskID	データ量
Reduce データ量	13	13_m_5	13_r_1	1000
Reduce データ量	13	13_m_5	13_r_2	1200
Reduce データ量	13	13_m_5	13_r_3	8000

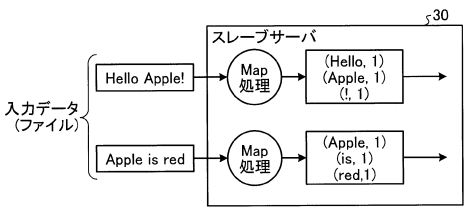
【図６】

実施例1に係るスレーブサーバの機能構成を示す機能ブロック図



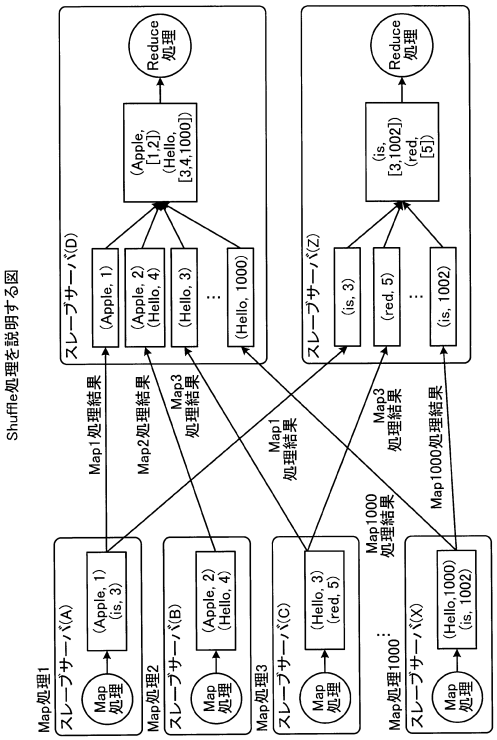
【図７】

Map処理を説明する図

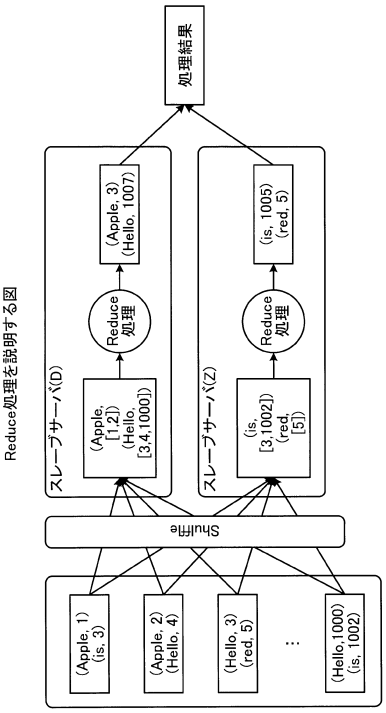


【図８】

Shuffle処理を説明する図



【図９】



【図１０】

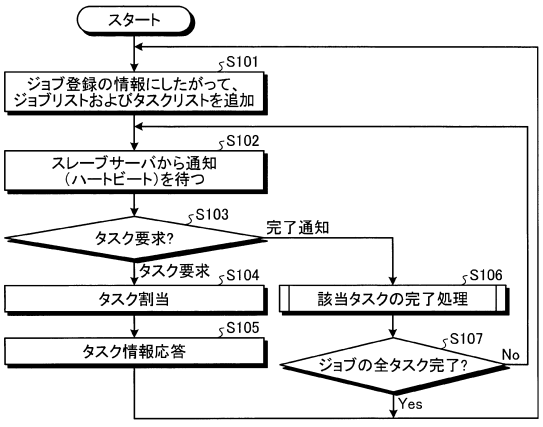
Reduce処理タスクの処理量を予測してフラグを設定する処理を説明する図

JobID	TaskID	種別	Reduceの順番	データのあるスレーブID	状態	割り当てスレーブID	必要スロット数	処理データ量	フラグ
1	1_m_1	Map	–	Node1, Node2	Done	Node1	1		
1	1_m_2	Map	–	Node3, Node4	Done	Node4	1		
1	1_m_3	Map	–	Node1, Node3	Running	Node1	1		
1	1_m_4	Map	–	Node2, Node4	Running	Node2	1		
1	1_r_1	Reduce	1	–	Running	Node2	1	12000	
1	1_r_2	Reduce	2	–	Running	Node3	1	25000	true
2	2_m_1	Map	–	Node3, Node4	Done	Node1	1		
2	2_m_2	Map	–	Node1, Node3	Done	Node2	1		
2	2_m_3	Map	–	Node2, Node4	Running	Node3	1		
2	2_m_4	Map	–	Node1, Node4	Running	Node4	1		
2	2_r_1	Reduce	1	–	Not assigned	–	1	24000	true
2	2_r_2	Reduce	2	–	Not assigned	–	1	13000	

S1 S2 S3 S4 S5

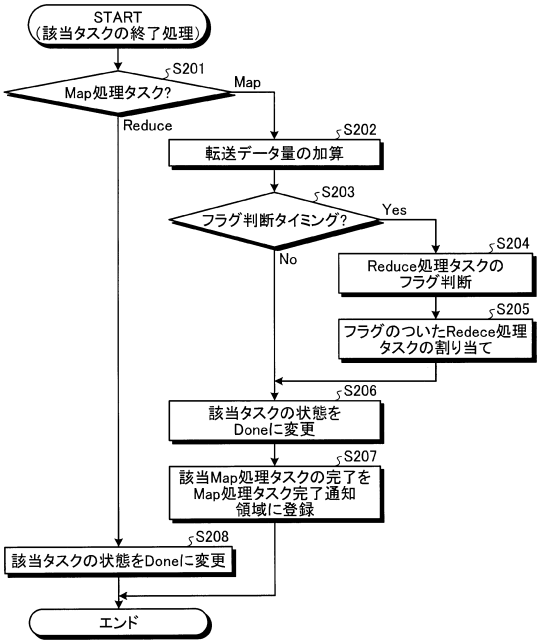
【図１１】

実施例1に係るマスタサーバが実行する処理の流れを示すフローチャート



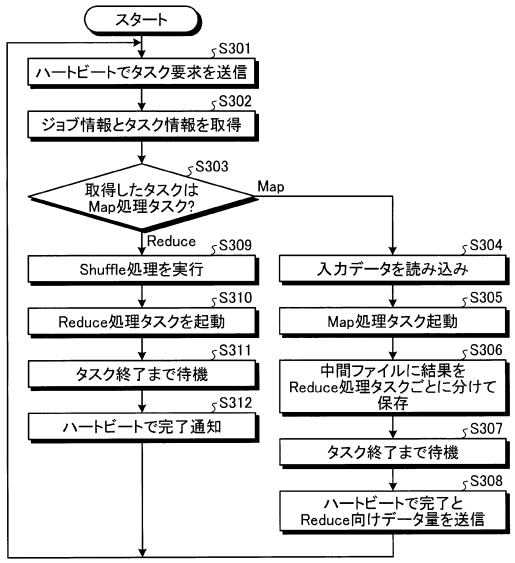
【図１２】

マスタサーバが実行する該当タスクの完了処理の流れを示すフローチャート



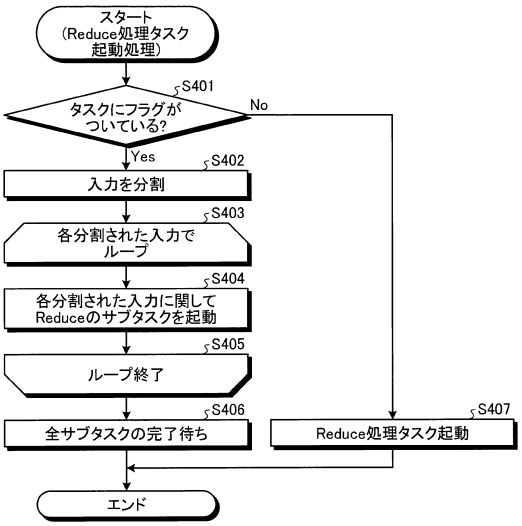
【図１３】

実施例1に係るスレーブサーバが実行する処理の流れを示すフローチャート



【図１４】

スレーブサーバが実行するReduce処理タスクの起動処理の流れを示す
フローチャート

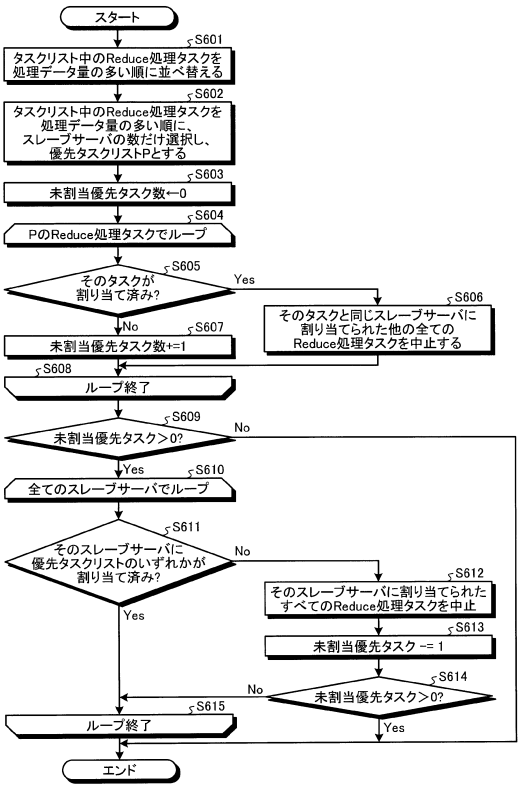


【図１５】

スレーブサーバが実行するReduce処理タスクの分割処理の流れを示す
フローチャート

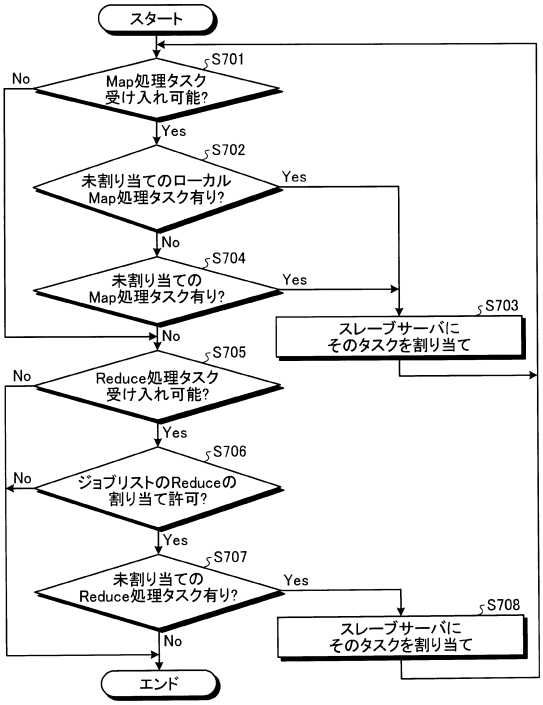
【図１６】

実施例2に係るReduce処理タスクの割当処理の流れを示すフローチャート



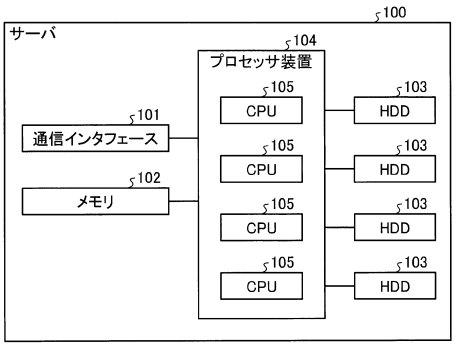
【図１７】

実施例3に係るReduce処理タスクの割当処理の流れを示すフローチャート



【図１８】

各サーバのハードウェア構成例を示す図



フロントページの続き

審査官　井上　宏一

(56)参考文献　特開２０１２－１１８６６９（ＪＰ，Ａ）
国際公開第２０１３／０２４５９７（WO，Ａ１）
特開２０１３－２５４２８０（ＪＰ，Ａ）
国際公開第２０１４／０２０７３５（WO，Ａ１）

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