(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号
特許第3681415号
(P3681415)
(45)発行日　平成17年8月10日(2005.8.10)　(24)登録日　平成17年5月27日(2005.5.27)

(51)Int.Cl.[7]　F I
G06F 9/46　G06F 9/46　340G
G06F 11/30　G06F 11/30　305G

請求項の数 8　(全 26 頁)

(21)出願番号　特願平6-61771
(22)出願日　平成6年3月30日(1994.3.30)
(65)公開番号　特開平6-337798
(43)公開日　平成6年12月6日(1994.12.6)
審査請求日　平成12年6月23日(2000.6.23)
審判番号　不服2004-13022(P2004-13022/J1)
審判請求日　平成16年6月24日(2004.6.24)
(31)優先権主張番号　特願平5-96928
(32)優先日　平成5年3月30日(1993.3.30)
(33)優先権主張国　日本国(JP)

(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　100090516
弁理士　松倉　秀実
(74)代理人　100113608
弁理士　平川　明
(74)代理人　100105407
弁理士　高田　大輔
(74)代理人　100089244
弁理士　遠山　勉
(72)発明者　藤田　和彦
神奈川県川崎市中原区上小田中１０１５番地　富士通株式会社内

最終頁に続く

(54)【発明の名称】デッドロック検出装置

(57)【特許請求の範囲】

【請求項１】

複数のタスクが共通の資源を利用するマルチタスクシステムにおいて前記複数のタスクが互いに占有している資源を待ち合って停止してしまうデッドロックを検出するためのデッドロック検出装置であって、複数のタスクを並列実行するために、前記タスクの実行を管理するタスク管理部と、各タスクがどの資源をロックしているかを管理するロック管理部と、一のタスクが他のタスクがロックしている資源を獲得要求した場合には、前記一のタスクが前記他のタスクを待っているとしてこの各タスクの「待ち関係」を登録する待ち管理テーブルと、前記ロック管理部と非同期で動作するとともに、前記待ち管理テーブルに登録された「待ち関係」からデッドロックを検出するデッドロック検出部とを備え、

前記マルチタスクシステムは、複数のシステムを有する分散システム上に実現され、各システムがそれぞれ前記タスク管理部、前記ロック管理部、前記待ち管理テーブル、前記デッドロック検出部を備え、

一方のシステムのタスクが他方のシステムのタスクに対して「待ち」の状態にあるときには、その「待ち関係」を前記一方のシステムの待ち管理テーブルに登録し、デッドロックを検出しないときは前記一方のシステムから前記他方のシステムに通信して前記他方のシステムの待ち管理テーブルに登録し、前記他方のシステムにおいてその待ち管理テーブルを見てデッドロックの有無を判定することを特徴とするデッドロック検出装置。

【請求項２】

前記ロック管理部には、各タスクとそれによりロックされている資源との関係が登録され

るロック管理テーブルを有していることを特徴とする請求項１記載のデッドロック検出装置。

【請求項３】

タスクにおいて「待ち関係」が発生したとき、前記デッドロック検出部は、前記待ち管理テーブルに「待ち関係」を登録するとともに、前記待ち管理テーブルを検索してデッドロックの有無を検出することを特徴とする請求項１記載のデッドロック検出装置。

【請求項４】

前記デッドロック検出部は、要求キュー受付部を有し、この要求キュー受付部で、前記デッドロック検出命令である「待ち関係の登録要求」を受け付けることを特徴とする請求項１記載のデッドロック検出装置。

【請求項５】

前記待ち関係にある２以上のタスクのそれぞれが別のシステム内にあるときには、各システムに設けた前記待ち管理テーブルに各タスクにおける「待ち関係」を登録するとともに、自己のシステムのタスクが他のシステムのタスクに対して「待ち」の状態にある場合には、各システムのデッドロック検出部は、その待ち先のシステムの待ち管理テーブルに通信でアクセスして自己のシステムの前記「待ち」の状態を示す「待ち関係」を送信するとともに、前記待ち先のシステムのデッドロック検出部に対して、自己のシステムの前記「待ち関係」と待ち先のシステムの待ち管理テーブルの登録内容とを突き合わせてデッドロックの有無を判定させることを特徴とする請求項１記載のデッドロック検出装置。

【請求項６】

前記マルチタスクシステムは、複数のシステムを有する分散システム上に実現され、あるタスクについて「待ち関係」が一定時間継続している場合に、そのタスクに対して再度資源獲得要求を出す待ち時間監視部を備えたことを特徴とする請求項１記載のデッドロック検出装置。

【請求項７】

複数のタスクが共通の資源を利用するマルチタスクシステムにおいて前記複数のタスクが互いに占有している資源を待ち合って停止してしまうデッドロックを検出するためのデッドロック検出装置であって、複数のタスクを並列実行するために、前記タスクの実行を管理するタスク管理部と、タスクとそれによりロックされている資源との関係を登録する第１のテーブルと、各タスクがどの資源をロックしているかを管理し、他のタスクによりロックされているものとして前記第１のテーブルに登録されている資源を一のタクスが獲得要求した場合には、前記一のタスクが前記他のタスクを待っている「待ち関係」の発生を検出するロック管理部と、前記待ち関係を登録する第２のテーブルと、前記ロック管理部とは非同期に動作するとともに、この第２のテーブルに登録された「待ち関係」から、デッドロックを検出するデッドロック検出部とを備え、

前記マルチタスクシステムは、複数のシステムを有する分散システム上に実現され、各システムがそれぞれ前記タスク管理部、前記ロック管理部、前記第２のテーブル、前記デッドロック検出部を備え、

一方のシステムのタスクが他方のシステムのタスクに対して「待ち」の状態にあるときには、その「待ち関係」を前記一方のシステムの第２のテーブルに登録し、デッドロックを検出しないときは前記一方のシステムから前記他方のシステムに通信して前記他方のシステムの第２のテーブルに登録し、前記他方のシステムにおいてその第２のテーブルを見てデッドロックの有無を判定することを特徴とするデッドロック検出装置。

【請求項８】

複数のタスクが共通の資源を利用するマルチタスクシステムにおいて前記複数のタスクが互いに占有している資源を待ち合って停止してしまうデッドロックを検出するためのデッドロック検出方法であって、

前記マルチタスクシステムは、複数のシステムを有する分散システム上に実現され、各システムがそれぞれタスク管理部、ロック管理部、待ち管理テーブル、デッドロック検出部を備え、

前記ロック管理部が、各タスクがどの資源をロックしているかを認識し、一のタスクが獲得要求している資源が既に他のタスクによってロックされているか否かを検知し、資源が既に他のタスクによってロックされていることを検知した場合には、前記一のタスクが前記他のタスクを待っていると認識し、この「待ち関係」を前記待ち管理テーブルに登録し、

前記デッドロック検出部が、登録された「待ち関係」が各タスクが互いに待ち合っていることを示す場から、デッドロックとして検出し、

一方のシステムのタスクが他方のシステムのタスクに対して「待ち」の状態にあるときには、その「待ち関係」を前記一方のシステムの待ち管理テーブルに登録し、デッドロックを検出しないときは前記一方のシステムから前記他方のシステムに通信して前記他方のシステムの待ち管理テーブルに登録し、前記他方のシステムにおいてその待ち管理テーブルを見てデッドロックの有無を判定することを特徴とするデッドロック検出方法。

【発明の詳細な説明】

【０００１】

【産業上の利用分野】

本発明はマルチタスクシステムにおけるデッドロックの検出装置に関する。

【０００２】

【従来の技術】

近年、コンピュータを用いた情報処理システムにおいて、複数のタスクあるいはトランザクションを同時に実行するマルチタスクシステムが発達してきた。タスクとは、ＣＰＵ内部における仕事の単位である。トランザクションとは、ひとつの完結したデータ操作を行うオペレーションの集まりである。マルチタスクとは、複数のプログラム（タスク，トランザクション）が、単一のコンピュータシステム，又は相互に情報交換可能に接続された複数のコンピュータシステム上で、同時に並行して実行される状態である。

【０００３】

このマルチタスクシステムでは、２以上のタスクが資源を共用する場合がある。その場合において、２以上のタスクの夫々が、そのタクスの実行に必要であり且つ他方のタクスの実行にも必要な複数の資源を、一部づつを占有（ロック）し合うケースが生じ得る。そのケースでは、お互いに他方のタクスが占有（ロック）している資源を待ち合うので、双方のタスクが停止し、それ以上プロセスを実行できない状態となってしまう。このような状態は、デッドロックの状態と呼ばれている。

【０００４】

図３に、デッドロックの状態の例を示す。図３の例は、２つのコンピュータシステムｉ，ｊから構成される分散システムにおける例を示している。一方のコンピュータシステムｉではタスクｘが実行され、他方のコンピュータシステムｊではタスクｙが実行されている。また、各コンピュータシステムｉ，ｊでアクセスできる資源として、Ａ，Ｂという２つの資源があるとする。なお、資源とは、タスクに割り当てられるプログラム，ファイル，データ等のソフトウェアを指す。ここでは、各コンピュータシステムｉ，ｊ外に存在するデータベースの中身（ページ，レコード等）として説明する。

【０００５】

図３において、タスクｘは資源Ａをロックしており、タスクｙはＢをロックしている。同時に、タスクｙは資源Ａをも必要としているので、資源Ａをロックすることを待っている。同様に、タスクｘは資源Ｂをも必要としているので、資源Ｂがロックできるようになることを待っている。この場合、タスクｘが資源Ａのロックを解除しない限り、タスクｙは資源Ａをロックできない。一方、タスクｙが資源Ｂのロックを解除しない限り、タスクｘは資源Ｂをロックできない。この結果、タスクｘ，ｙは互いがロックしているＡ，Ｂを待ち合って停止する。両タスクｘ，ｙが停止すると、各々が既にロックしている資源Ａ，Ｂの解除もできなくなってしまうので、この状態は永遠に続くことになる。よって、各タスクはそれ以上のプロセスを実行できない。

【０００６】

このようなデッドロックは、コンピュータシステムがマルチプロセッサ方式のシステムであるかシングルプロセッサ方式のシステムであるか，あるいは、コンピュータシステムがスタンドアローンで運用されるのか分散処理システムを構成するのかに拘らず、システムがマルチタスクシステムであれば生じ得る問題である。

【０００７】

このようなデッドロックが生じたとき、これを修復する手段を講じなければならない。そのためには、前提としてデッドロックが生じたことを検出しなければならない。

【０００８】

デッドロック検出には、実用性を向上させる理由から、以下のスペックを満たすことが要求される。

第１に、実際はデッドロックではないにも拘らずデッドロックと誤認してしまう現象，すなわち疑似デッドロック（ｐｈａｎｔｏｍ　ｄｅａｄｌｏｃｋ）の検出が防止されていなければならない（第１の要求）。

【０００９】

第２に、全てのデッドロックが検出されなければならない。換言すれば、現実にデットロックが生じているに場合には、デッドロックを検出できるときと検出できないときがあってはならず、全てデッドロックであると検出されなければならない（第２の要求）。

【００１０】

第３に、デッドロック検出を行うことによるシステムへの影響を小さく抑えなければならない。即ち、デッドロックを検出するためにタスクを停止するようなことは、できるだけ避けなければならない（第３の要求）。

【００１１】

なお、マルチタスクシステムを分散処理システム上で実現する場合には、上記各要求の他に次のスペックが要求される。即ち、デッドロックを検出するためにシステム間で通信を行う必要があるが、この通信のオーバーヘッドをできるだけ削減しなければならない（第４の要求）。

【００１２】

従来のデッドロック検出装置では、以下のようなような条件を満足させることによって、上述した第１乃至第３の要求を満足してデッドロックを検出しようとしていた。その条件とは、

(a)　トランザクションの非同期ａｂｏｒｔ（異常終了）が発生しないこと，

(b)　マルチタスクシステムを分散処理システム上で実現する場合には、各システム間の通信メッセージの遅延・消失が発生しないこと，

(c)　デッドロック検出中のトランザクション待ち関係の変更がないこと。

【００１３】

(d)　マルチタスクシステムを分散処理システム上で実現する場合には、システムの非同期ダウンが発生していないこと，である。

【００１４】

上述の第１乃至第３の要求と(a)乃至(d)の条件との関係の関係を説明する。

(a)の条件に関し、トランザクションの非同期ａｂｏｒｔ（異常終了）が発生すると、前記第１の要求における疑似デッドロック（ｐｈａｎｔｏｍ　ｄｅａｄｌｏｃｋ）の検出防止を図ることができない。例えば、タスク（トランザクション）ｘが資源Ａをロックしており、タスク（トランザクション）ｙが資源Ｂをロックしている場合、タスクｙが資源Ａを待ち、タスクｘが資源Ｂに待ち要求を出した時点で、タスクｙが非同期に異常終了してしまったとする。この場合、タスクｙの非同期終了によってタスクｙによる資源Ｂのロックが解除されるので、タスクｘは資源Ｂをロックできる。従って、本来ならばデッドロックは発生しないはずである。しかし、タスクｙの非同期異常終了による資源Ｂのロック解除は直ちに検出できないので、現実には、デッドロックが発生していないにもかかわらずデッドロックが発生したものとして扱われてしまう。

【００１５】

(b)の条件に関し、システムの非同期ダウンが生じると、デッドロックを検出できるときと検出できないときが生じ、前記第２の要求における全デッドロックの検出を行うことができない。なぜならば、分散処理システムにおいてはシステム間の通信によって共通資源へのアクセスをするわけであるが、その通信メッセージの伝達が遅れる場合や通信異常により消失する場合があると、デッドロックが発生したこと自体不明となるからである。

【００１６】

同様に、(d)の条件に関し、システム間の通信メッセージの遅延・消失が生じると、デッドロックを検出できるときと検出できないときが生じ、前記第２の要求における全デッドロックの検出を行うことができない。なぜならば、一方のシステム作動中に他方のシステムがダウンしてしまうと、システムにおけるタスクに関する管理情報が失われ、デッドロック検出の判定ができなくなるからである。

【００１７】

さらに、(c)の条件に関し、デッドロック検出中にトランザクション待ち関係の変更があると、現実にどのタスクがどの資源をロックしているかに関する情報が混乱してしまうので、第１又は第２の要求を満たすことができない。

【００１８】

【発明が解決しようとする課題】

しかしながら、(c)の条件は、デッドロック検出中における新たなトランザクション（タスク）の発生や待ち関係の発生を全て禁止することを内容とするものである。すなわち、デッドロック検出のためには、システムにおける要求受付を一旦停止する必要があるとする条件である。従って、本来デッドロックに関係ない資源（その資源を仮に資源Ｚとする。）に要求を出しているタスクがあっても、その要求を停止しなければならないことになる。従って、この条件(c)を追求すると、かえってシステムの円滑な運用が図れなくなり、第３の要求を満足できない結果となる。

【００１９】

なお、条件(a) に起因する疑似デッドロックの検出を防ぐことは、現実には不可能である。すなわち、各システムにどのような異常が生ずるかを予想しこれをすべて回避することは不可能であって、タスクが非同期に異常終了することは防止することはできないからである。

【００２０】

そこで、本発明の第１の技術的課題は、以上の問題点に鑑み、デッドロック検出中のトランザクション待ち関係の変更があってもデッドロック検出を継続でき、それによりデッドロック検出を行うことによるシステムへの影響を小さくすることができるデッドロック検出装置を提供することである。

【００２１】

なお、本発明の第２の技術的課題は、分散処理システムを対象としたデッドロック検出装置において、システム間の通信メッセージの遅延・消失が発生した場合，デッドロック検出中のトランザクション待ち関係の変更が有った場合，及びシステムの非同期ダウンが生じた場合の何れにおいても、全てのデッドロックを検出でき、疑似デッドロックを検出せず、通信のオーバーヘッドをできるだけ削減でき、デッドロック検出を行うことによるシステムへの影響を小さくすることができるデッドロック検出装置を提供することをである。

【００２２】

【課題を解決するための手段】

本発明は、前記第１の課題を解決するために、図１の原理図のように、以下の手段を採用した。

【００２３】

＜本発明の要旨＞

即ち、複数のタスク１００が共通の資源１０１を利用するマルチタスクシステムにおいて前記複数のタスク１００が互いに占有している資源１００を待ち合って停止してしまうデ

ッドロックを検出するためのデッドロック検出装置であって、複数のタスク１００を並列実行するために、前記タスク１００の実行を管理するタスク管理部（ＴＭ）１０２と、各タスクがどの資源１００をロックしているかを管理するロック管理部（ＬＭ）１０３と、一のタスクが他のタスクがロックしている資源を獲得要求した場合には、前記一のタスクが前記他のタスクを待っているとしてこの各タスクの「待ち関係」を登録する待ち管理テーブル（ＬＴ）１０５と、前記ロック管理部（ＬＭ）１０３と非同期で動作するとともに、前記待ち管理テーブル（ＬＭ）１０３に登録された「待ち関係」からデッドロックを検出するデッドロック検出部（ＤＤ）１０４とを備えたことを特徴とする。

【００２４】

以下に、本発明の構成要素の概要と、そのポイントを簡単にまとめる。

【００２５】

〔タスク〕

“タスク”とは、通常ＣＰＵ内部における仕事の単位を意味する。本発明においては、“タスク”を“トランザクション”と言い替えることができる。この“トランザクション”とは、ひとつの完結したデータ操作を行うオペレーションの集まりを意味し、“タスク”に含まれる概念であり、プログラムによって実行されるものである。要するに、本発明は、複数のプログラムが同時に並行して実行されるとき、各プログラムが資源を共有してロック状態となるのを検出しようとするものである。よって、“タスク”との用語を用いても、“トランザクション”との用語を用いても、単にプログラムの実行単位との用語を用いても、本発明においては、用語の差異は特に問題とはならない。以下、“タスク”＝“トランザクション”と理解しても本発明の実施において何等の問題もない。また、本発明において、各タスクが共有する資源とは、データの集合であるファイルやファイルの中の下層的に記録されたレコードなどである。本発明でロックとは、或るタスク又はトランザクションがファイル全体を占有すること、あるいは、ファイルの下の或るレコードを占有することをいう。

【００２６】

〔デッドロック検出〕

デッドロック検出部（ＤＤ）１０４におけるデッドロック検出は、例えば、次の通りにすることができる。即ち、タスク（トランザクション）と資源の占有関係，即ち「待ち関係」を前記待ち管理テーブル（ＬＴ）１０５により登録する。この待ち管理テーブル（ＬＴ）１０５をデッドロック検出部（ＤＤ）１０４が見て、デッドロックを検出する。この検出は、前記ロック管理部（ＬＭ）１０３によるロック管理とは別個に行われる。好ましくは、前記ロック管理部（ＬＭ）１０３が、「あるタスクがある資源について「待ち関係」となった」ことを検出したとき、デッドロック検出部（ＤＤ）１０４に「待ち関係」を待ち管理テーブル（ＬＴ）１０５に登録するよう要求する。そして、その登録内容を参照することでデッドロックの有無を判定する。

【００２７】

デッドロック検出のためにトランザクションの待ち関係を待ち管理テーブル（ＬＴ）１０５に登録する方法としては、以下の方法が好適である。即ち、トランザクションの待ち関係をグラフによって表現する。このグラフを、ここではウェイトフォーグラフ（ＷＦＧ：Wait-forーgraph）と呼ぶ。このグラフを前記待ち管理テーブル（ＬＴ）１０５に登録するのである。

【００２８】

このグラフにおいて、システムｉで発生したトランザクションｘをＴ（ｉ，ｘ）で定義し、システムｊで発生したトランザクションｙをＴ（ｊ，ｙ）で定義する。また、Ｔ（ｉ，ｘ）がＴ（ｊ，ｙ）について待つこと，即ちＴ（ｊ，ｙ）がロックしている資源を解放するのをＴ（ｉ，ｘ）が待つことを、

Ｔ（ｉ，ｘ）→Ｔ（ｊ，ｙ）

と表すこととする。この場合、Ｔ（ｊ，ｙ）が終了しない限り、Ｔ（ｉ，ｘ）はそれ以上プロセスを進めることができない。

【００２９】
Ｔ（ｉ，ｘ）→Ｔ（ｊ，ｙ）
と
Ｔ（ｊ，ｙ）→Ｔ（ｉ，ｘ）
とが同時に成立したときには、
Ｔ（ｉ，ｘ）→Ｔ（ｊ，ｙ）→Ｔ（ｉ，ｘ）
というループが形成される。この場合はデッドロック状態であるので、このループを検出することにより、デッドロックの検出をすることができる。
【００３０】
ところで、以上は２つのシステム間でのデッドロック検出例の説明であるが、自システム内でのデッドロックは、
Ｔ（ｉ，ｘ）→Ｔ（ｉ，ｙ）→Ｔ（ｉ，ｘ）
で表現される。
【００３１】
本発明の特徴点は、タスク管理部（ＴＭ）１０２やロック管理部（ＬＭ）１０３等，タスクの実行に必要なブロックから独立したデッドロック検出部（ＤＤ）１０４をシステム（ｉ，ｊ）内に設け、このロック管理部（ＬＭ）１０３とは非同期にデッドロック検出部（ＤＤ）１０４を作動させる点にある。
【００３２】
従来のデッドロック検出方法においては、デッドロックを検出するために、各タスクの実行を一旦停止させていた。そして、その間に、ロック管理部（ＬＭ）１０３におけるロック情報に基づいてデッドロックの有無を判定していた。しかしながら、これでは、タスクの円滑な実行を確保できない。
【００３３】
これに対し、本発明では、タスク管理部（ＴＭ）１０２やロック管理部（ＬＭ）１０３から分離した待ち管理テーブル（ＬＴ）を設けて、前記ロック管理部（ＬＭ）１０３からのロック情報を待ち管理テーブル（ＬＴ）１０５に登録しておく。そして、タスクの実行とは独立して作動するデッドロック検出部（ＤＤ）１０４を設けた。このデッドロック検出部（ＤＤ）１０４は、あるタスクが資源を待つ状態に入ったという情報をロック管理部（ＬＭ）１０３が受けた時、そのタスクの待ちの関係も踏まえて、前記待ち管理テーブル（ＬＴ）１０５の登録内容を見てデッドロックの有無を判定するように構成することができる。
【００３４】
このように、本発明は、タスク管理部（ＴＭ）１０２やロック管理部（ＬＭ）１０３等，タスクの実行に必要なシステムから分離して、デッドロック検出部（ＤＤ）１０４を設け、独立して作動させるため、デッドロック検出の為にタスクの実行を停止する必要がない。
【００３５】
ところで、デッドロック検出部（ＤＤ）１０４によるデッドロック検出は、前記ロック管理部（ＬＭ）１０３によってタスクの待ち関係が検出された時に行うのが好ましい。すなわち、ロック管理部（ＬＭ）１０３によりタスクの待ち関係が検出された場合には、デッドロック検出部（ＤＤ）１０４がその待ち関係を管理テーブル（ＬＴ）１０５に登録する。この登録は、デッドロック検出の開始用トリガーとなる。デッドロック検出部（ＤＤ）１０４は、この登録の通知を受けることを契機に、待ち管理テーブル（ＬＴ）１０５を参照してデッドロックの有無検出を行う。
【００３６】
デッドロックが検出されたとき、いずれかのタスクを強制的に異常終了させなければデッドロックを修復できない。いずれのタスクを強制的に異常終了させるかはシステムにより異なる。例えば、タスクの開始時刻の遅い方が仕事量が少ないとみてそのタスクを終了させても良い。あるいは、仕事量を実際に計上して少ない仕事量のタスクを終了させるよう

にしても良い。
【００３７】
＜分散システムへの適用＞
本発明によるデッドロック検出装置は、複数のシステムを有する分散システム上に実現することができる。この様な分散システムを採用する場合には、上記した第１の課題に加えて、第２の課題の達成を考慮しなければならない。この場合のデッドロック検出は以下の様になる。
【００３８】
即ち、前記待ち管理テーブル（ＬＴ）１０５に前記「待ち関係」を登録する場合において、２以上のタスク（ｘ，ｙ）が同一システム内のものであれば、そのシステムに設けた待ち管理テーブル（ＬＴ）１０５に各タスクにおける「待ち関係」を登録すれば足りる。
【００３９】
一方、あるシステムのタスクが他のシステムのタスクに対して「待ち」の状態にある場合には、その「待ち関係」を一方のシステムから他方のシステムの待ち管理テーブル（ＬＴ）１０５へ通知すれば良い。この通知を受けた待ち管理テーブル（ＬＴ）１０５は、この「待ち関係」を登録する。この登録と同時に、他方のシステムのデッドロック検出部（ＤＤ）１０４がその待ち管理テーブル（ＬＴ）１０５を見に行き、デッドロックの有無を判定することができる。逆に、「待ち関係」を他方のシステムから一方のシステムの待ち管理テーブル（ＬＴ）１０５へ通知すれば、一方のシステムで、その待ち管理テーブル（ＬＴ）１０５を見てデッドロックの有無を判定できる。以上は、自己のシステムの待ち管理テーブル（ＬＴ）１０５に、自己のシステムのタスクの「待ち関係」と、その待ち先のシステムのタスクの「待ち関係」を両方とも登録する場合のことである。
【００４０】
これとは別に、自己のシステムの「待ち関係」のみを自己の管理テーブル（ＬＴ）１０５に登録するようにしても良い。この場合には、デッドロック検出をする際に、待ち先のシステムの待ち管理テーブル（ＬＴ）１０５に通信でアクセスする。そして、自己のシステムのタスクの「待ち関係」と、その待ち先のタスクの「待ち関係」とを突き合わせ、上述したループが形成されていればデッドロックとして検出することができる。
【００４１】
本発明を分散システムに適用した場合、自己システム内でのデッドロックに対しては、複数のシステム相互間でデッドロック検出のための情報の通信は行わない。即ち、複数のシステム間でデッドロックが発生した場合のみ情報の通信を行う。但し、デッドロックは二者間で生じることがほとんどであるので、１回の通信でデッドロックを検出できる。従って、上記第２の課題を達成することができる。
【００４２】
＜待ち時間管理テーブルの付加＞
上記した本発明の必須の構成要件に、図２の原理図に示すように、待ち時間監視部（ＷＴ）１０６を設けてもよい。この待ち時間監視部（ＷＴ）１０６は、あるタスク（トランザクション）について「待ち関係」が一定時間継続している場合に、そのタスク（トランザクション）について再度資源獲得要求を出すブロックである。この待ち時間監視部（ＷＴ）１０６を設ける目的は、以下に説明する通りである。
【００４３】
即ち、デッドロックが生じても、デッドロックを検出できないと、デッドロックを修復できない。デッドロックが発生しているにも拘らずデッドロックを検出することができない原因としては、通信の欠落により管理テーブル（ＬＴ）１０５に「待ち関係」の情報が登録されていないことが考えられる。そこで、あるタスクにつき「待ち関係」が一定時間継続している場合には、待ち時間監視部（ＷＴ）１０６が再度資源獲得要求を出すようにするのである。これにより、「待ち関係」の情報をその待ち先のシステムの待ち管理テーブル（ＬＴ）１０５に再度送信する契機を与えることができる。従って、確実にデッドロックを検出することができる。

【００４４】
【作用】
本発明によるデッドロック検出装置では、タスクの実行状況は、タスク管理部（ＴＭ）１０２によって管理される。この際、各タスクが資源を占有する場合には、ロック管理部（ＬＭ）１０３によって、どのタスクがどの資源を占有したかについての情報が管理される。そして、他のタスクが占有している資源を一のタスクが獲得要求すると、この一のタスクは他のタスクの終了を待たねばならない。この「待ち関係」は、待ち管理テーブル（ＬＴ）１０５において登録管理される。
【００４５】
この待ち管理テーブル（ＬＴ）１０５をデッドロック検出部（ＤＤ）１０４が見て、デッドロックを検出する。この検出は、ロック管理部（ＬＭ）１０３によるロック管理とは別個に行われる。従って、デッドロック検出部（ＤＤ）１０４によるデッドロックが行われていても、ロック管理部（ＬＭ）１０３はその動作を行うことができる。よって、新たなタスクの発生や待ち関係の発生を禁ずる必要がなくなる。そのため、デッドロック検出を行うことによるシステムへの影響を小さくすることができるのである。
【００４６】
【実施例】
以下、本発明の好適実施例を、図面を参照して説明する。ここでは、今まで使用した“タスク”という言葉を“トランザクション”で置き換えて説明する。また、この好適実施例は、本発明を分散処理システムにおいて実施する場合の具体例である。
【００４７】
＜システムの概要＞
図４には分散処理システムの構成が示されている。この分散処理システムにおいては、二つのコンピュータシステム（システムｉ及びシステムｊ）が分散して設けられ、相互にネットワーク（ＮＷ）３０によって接続されている。また、両コンピュータシステム（ｉ，ｊ）とネットワーク（ＮＷ）３０によって接続され、且つ両コンピュータシステム（ｉ，ｊ）からアクセス可能なデータベース（ＤＢ）２０が設けられている。このようなシステムは、例えば、預金システムに利用される。
【００４８】
図４から明かなように、各コンピュータシステム（システムｉ及びシステムｊ）は、トランザクション管理部（ＴＭ）１０，資源管理部（ＲＭ）１１，ロック管理部（ＬＭ）１２，デッドロック検出部（ＤＤ）１５，待ち管理テーブルＴ３，及びウォッチドックタイマ（ＷＴ）１３を備えている。なお、システムｊはシステムｉと全く同じ構成を有している。そのため、図４においては、システムｉについてのみその詳細な構成を示し、システムｊについてはその詳細な構成の図示を省略した。
【００４９】
データベース（ＤＢ）２０には、資源としてのファイル又はレコードが複数個格納されている。図４においては、これら資源として、資源Ａ及び資源Ｂを例示した。
【００５０】
以下、各構成ブロックを詳細に説明する。
【００５１】
＜トランザクション管理部（ＴＭ）＞
トランザクション管理部（ＴＭ）１０は、複数のトランザクションの実行を管理している。トランザクション管理部（ＴＭ）１０を、タスク管理部（ＴＭ）１０と言っても良い。
【００５２】
このトランザクション管理部（ＴＭ）１０は、応用プログラムからのトランザクション開始・正常終了（commit）・異常終了（abort）の通信を受付け、システム内でのトランザクションを管理するブロックである。より詳しく言うと、例えばシステムｉにおいてトランザクションｘが開始されたときには、Ｔ（ｉ，ｘ）という形式のデータを登録し、トランザクションｘが終了・異常終了したときにはこのＴ（ｉ，ｘ）という形式のデータを削

除するのである。

【００５３】

トランザクション管理部（ＴＭ）１０は、トランザクションから資源の要求を受け付けると、その要求を資源管理部（ＲＭ）１１に渡し、その応答（ｏｋ／ｎｏ）をもらう。また、トランザクション管理部（ＴＭ）１０は、デッドロック検出部（ＤＤ）１５から送信されたトランザクションのデッドロック通知を受け付けて、トランザクションを終了させる。また、トランザクション管理部（ＴＭ）１０は、デッドロック検出部（ＤＤ）１５から送信されたリトライ通知を受け付けて、資源管理部（ＲＭ）１１に資源獲得要求を再発行する。また、トランザクション管理部（ＴＭ）１０は、他のコンピュータシステム（システムｊ）のトランザクションの正常終了や異常終了の通信を受信し、デッドロック検出部（ＤＤ）１５にグラフの登録・削除を要求する。

【００５４】

図４において、“ｓｔａｒｔ”はトランザクションの実行開始を意味し、“ａｂｏｒｔ”はトランザクションの異常終了を意味し、“ｃｏｍｍｉｔ”はトランザクションの正常終了を意味する。

【００５５】

トランザクション管理部（ＴＭ）１０における、資源獲得要求・資源解放要求は、二相ロック（２ＰＬ：Ｔｗｏ　Ｐｈａｓｅ　Ｌｏｃｋ）方式で実行される。これは、疑似デッドロック（ｐｈａｎｔｏｍ　ｄｅａｄｌｏｃｋ）の検出を防止するのに効果的である。

【００５６】

疑似デッドロックは、一般にグラフの登録と削除が競合した場合に発生する。例えば、

Ｔ（ｉ，ｘ）→Ｔ（ｊ，ｙ）

のグラフが既に登録されている場合において、

Ｔ（ｉ，ｘ）→Ｔ（ｊ，ｙ）

の削除要求と

Ｔ（ｊ，ｙ）→Ｔ（ｉ，ｘ）

の登録要求とが、デッドロック検出部（ＤＤ）１５に対して同時に発生したとする。この際、削除要求が先に受理された場合にはデッドロックが発生しない。これに対して、登録要求が先に受理された場合には、疑似デッドロックとなる。

【００５７】

Ｔ（ｉ，ｘ）→Ｔ（ｊ，ｙ）

の削除要求が発生するのは、このグラフで表される待ち関係がなくなった場合である（即ち、Ｔ（ｊ，ｙ）が資源のロックを解除した場合である。）。ロックの解除は、トランザクションが自ら資源のロックを解除する場合か非同期にａｂｏｒｔ（異常終了）する場合に行われる。

【００５８】

二相ロック方式は、ある処理がデータのロック（占有）を始めたらロックし続け、ロックを解除し始めたら解除し続けるという２つの相（フェーズ）からなるロック方式である。この方式によれば、複数のタスクやトランザクションがそれぞれ逐次実行されたのと同一結果となる。

【００５９】

トランザクションが非同期にａｂｏｒｔ（異常終了）しないとすれば、この方式により、一旦ロックが解除されると、新たなロック獲得の要求が発生しないことを保証する。つまり、一旦発生したグラフ

Ｔ（ｉ，ｘ）→Ｔ（ｊ，ｙ）

は、Ｔ（ｊ，ｙ）が終了するまで削除されることはない。従って、上記の

Ｔ（ｉ，ｘ）→Ｔ（ｊ，ｙ）

の削除要求が発生した後においては、Ｔ（ｊ，ｙ）が終了しているので、Ｔ（ｊ，ｙ）が新たな資源獲得要求をすることはない。よって、

Ｔ（ｊ，ｙ）→Ｔ（ｉ，ｘ）

の登録要求が発生しないことを保証できる。
【００６０】
以上の理由により、この方式を採用することによって、疑似デッドロックの発生原因をトランザクションの非同期ａｂｏｒｔ（異常終了）のみに限定することができる。
【００６１】
＜資源管理部（ＲＭ）＞
資源管理部（ＲＭ）１１は、このトランザクション管理部（ＴＭ）１０に双方向で接続されている。資源管理部（ＲＭ）１１は、資源管理デーブルＴ１を有している。資源管理部（ＲＭ）１１は、トランザクション管理部（ＴＭ）１０からの資源獲得要求及び資源解放要求の内容に基づいて、トランザクションとそのトランザクションが要求している資源との対応関係を、資源管理テーブルＴ１上にマッピングして管理している。
【００６２】
また、資源管理部（ＲＭ）１１は、トランザクション管理部（ＴＭ）１０からの資源獲得要求に応じてロック要求をロック管理部（ＬＭ）１２に対して行い、トランザクション管理部（ＴＭ）１０からの資源解放要求に応じて、ロック解放要求をロック管理部（ＬＭ）１２に対して行う。
【００６３】
＜ロック管理部（ＬＭ）＞
ロック管理部（ＬＭ）１２は、資源管理部（ＲＭ）１１に双方向で接続されている。ロック管理部（ＬＭ）１２は、ロック管理テーブルＴ２を有している。即ち、ロック管理部（ＬＭ）１２は、このロック管理テーブルＴ２によりロック状態の管理を行う制御部である。
【００６４】
トランザクションｘ，ｙと資源Ａ、Ｂがある場合において、トランザクションｘが資源Ａをロック（占有）し、トランザクションｙがＢをロックしたときには、ロック管理部（ＬＭ）１２は、この関係をロック管理テーブルＴ２に登録する。即ち、図４に示すように、ｘがＡをロックした状態を例えば（ｘ：Ａ）と定義し、ｙがＢをロックした状態を例えば（ｙ：Ｂ）と定義し、この情報をロック管理テーブルＴ２に登録する。
【００６５】
なお、このロック管理テーブルＴ２は、そのコンピュータシステム（ｉ又はｊ）におけるトランザクションについてのロック情報を管理するばかりでなく、他のコンピュータシステム（ｊ又はｉ）におけるトランザクションについてのロック情報をも管理する。この他のシステムにおけるトランザクションについてのロック情報は、コンピュータシステム間で通信を行うことにより獲得することができる。但し、各コンピュータシステム（ｉ，ｊ）によって共用される共用メモリ上に単一のロック管理テーブルＴ２を作成し、全コンピュータシステム（ｉ，ｊ）における全トランザクションに関するロック情報を一括管理させれば、各コンピュータシステム間における通信の必要はなくなる。
【００６６】
いま、上述した状態において、更に資源管理部（ＲＭ）１１から、トランザクションｙによる資源Ａのロック要求がなされ、トランザクションｘによる資源Ｂのロック要求がなされるとする。そうすると、ロック管理部（ＬＭ）１２はロック管理テーブルＴ２の情報を参照し、このようなロックができないことを認識する。この場合には、トランザクションｙはトランザクションｘによる資源Ａのロック解放を待ち、トランザクションｘはトランザクションｙによる資源Ｂのロック解放を待たねばならない。この待ち状態は、それぞれ、（ｘ→Ｂ）、（ｙ→Ａ）と定義される。このような定義が発生したとき、ロック管理部（ＬＭ）１２は「待ち」が発生したと判断するのである。
【００６７】
本実施例では、このような「待ち関係」を、ロック管理部（ＬＭ）１２とは切り離して、待ち管理テーブルＴ３に登録して管理する。即ち、「待ち関係」が生じたときには、ロック管理部（ＬＭ）１２は、上記定義に基づいて、ウェイトフォーグラフ登録をデッドロッ

ク検出部（ＤＤ）１５に要求する。この要求の際には、上記定義におけるトランザクション（ｘ，ｙ）がどのコンピュータシステムにおけるトランザクションであるのか、及び、上記定義における資源（Ａ，Ｂ）が現在どのコンピュータシステムのどのトランザクションによってロックされているのかの情報も、デッドロック検出部（ＤＤ）１５に通知する。

【００６８】

ロック管理部（ＬＭ）１２は、資源管理部（ＲＭ）１１からのロック要求が「待ち」にならない場合には、資源管理部（ＲＭ）１１に対してすぐに応答（ｏｋ）を返す。ロック管理部（ＬＭ）１２が判断して「待ち」が発生した場合のみ、待ち関係を示すウェイトフォーグラフを、待ち管理テーブルＴ３に登録する登録要求キューを発行する。従って、「待ちが発生しない資源獲得要求」に関しては、デッドロック検出中か否かに拘らず、その資源獲得要求を行ったトランザクションの実行処理は停止されない。

【００６９】

＜デッドロック検出部（ＤＤ）＞

デッドロック検出部（ＤＤ）１５は、待ち管理テーブルＴ３の登録内容からデッドロックの有無を判定する部分である。

【００７０】

デッドロック検出部（ＤＤ）１５は、要求キュー受付部（ＱＲ）１４を有する。この要求キュー受付部（ＱＲ）１４は自システムのロック管理部（ＬＭ）１２から待ち関係（ウェイトフォーグラフ）の登録・削除要求キューを受け付ける。また、他のコンピュータシステムからの待ち関係（ウェイトフォーグラフ）の登録・削除要求キューを受け付ける。さらに、他システムのトランザクションが異常終了（ａｂｏｒｔ）又は正常終了（ｃｏｍｍｉｔ）した場合には、トランザクション管理部（ＴＭ）１０からの待ち関係（ウェイトフォーグラフ）の削除要求キューを受け付ける。

【００７１】

デッドロック検出部（ＤＤ）１５は、これら要求キューに従い、先ずウェイトフォーグラフの登録又は削除を、待ち管理テーブルＴ３に対して行う。このウェイトフォーグラフ（Ｗａｉｔ・ｆｏｒ・ｇｒａｐｈ）の形式は以下の通りである。即ち、例えば、

システムｉで発生したトランザクションｘ＝Ｔ（ｉ，ｘ）、

システムｊで発生したトランザクションｙ＝Ｔ（ｊ，ｙ）

としたとき、　Ｔ（ｉ，ｘ）がＴ（ｊ，ｙ）について待つことを

Ｔ（ｉ，ｘ）→Ｔ（ｊ，ｙ）と表す。

【００７２】

ウェイトフォーグラフの登録を行う際には、デッドロック検出部（ＤＤ）１５は、通知された情報に基づいて、予めウェイトフォーグラフを作成する。

ウェイトフォーグラフの登録がなされると、デッドロック検出部（ＤＤ）１５はデッドロック検出を開始する。デッドロックが検出されたときは、デッドロック検出部（ＤＤ）１５は、トランザクション管理部（ＴＭ）１０にデッドロック通知を行う。

ウェイトフォーグラフの削除要求を受け付けた場合は、当該ウェイトフォーグラフを削除し、動作できるトランザクションに対しリトライ通知を行う。

【００７３】

＜待ち管理テーブルＴ３＞

待ち管理テーブルＴ３には、ウェイトフォーグラフが登録される。上述した通り、システムｉで発生したトランザクションｘ（資源Ａを占有中）＝Ｔ（ｉ，ｘ），システムｊで発生したトランザクションｙ（資源Ｂを占有中）＝Ｔ（ｊ，ｙ）としたとき、Ｔ（ｉ，ｘ）がＴ（ｊ，ｙ）について待つことを

Ｔ（ｉ，ｘ）→Ｔ（ｊ，ｙ）

と表す。この場合、トランザクションｙが占有している資源Ｂを更新してＴ（ｊ，ｙ）が終了しない限り、トランザクションｘは獲得しようとしている資源Ｂを使用できない。この状態を「待ち関係」といい、「Ｔ（ｉ，ｘ）がＴ（ｊ，ｙ）について待つ」という。

【００７４】
デッドロック検出部（ＤＤ）１５は、待ち管理テーブルＴ３に、この
Ｔ（ｉ，ｘ）→Ｔ（ｊ，ｙ）
のグラフを、「待ち関係」として登録する。
【００７５】
一方、この
Ｔ（ｉ，ｘ）→Ｔ（ｊ，ｙ）
のグラフの成立と同時に
Ｔ（ｊ，ｙ）→Ｔ（ｉ，ｘ）
が成立していることがある。この場合、トランザクションｘが占有している資源Ａを更新してＴ（ｉ，ｘ）が終了しない限り、トランザクションｙは獲得しようとしている資源Ａを使用できない。この２つの待ち関係を突き合わせると、
Ｔ（ｉ，ｘ）→Ｔ（ｊ，ｙ）→Ｔ（ｊ，ｙ）→Ｔ（ｉ，ｘ）
というループが形成される。よって、このループが検出されればデッドロックが発生しているということができるのである。
【００７６】
この待ち関係は、システムｉのトランザクションｘとシステムｊのトランザクションｙとの間で生じている。そして、
Ｔ（ｉ，ｘ）→Ｔ（ｊ，ｙ）
のグラフはシステムｉの待ち管理テーブルＴ３に登録され、
Ｔ（ｊ，ｙ）→Ｔ（ｉ，ｘ）
のグラフはシステムｊの待ち管理テーブル（Ｔ３）に登録される。このため、両者を突き合わせるためには、いずれかを他方に送信しなければならない。ここでは、「待ち関係」が登録されるとき、その待ち先に「待ち関係」を送信する。
【００７７】
すなわち、
Ｔ（ｉ，ｘ）→Ｔ（ｊ，ｙ）
がシステムｉの待ち管理テーブル（Ｔ３）に登録されたとき、システムｉのデッドロッ検出部（ＤＤ）１５は、システムｊの待ち管理テーブルＴ３に同一の内容のグラフを送信して登録する。
【００７８】
これにより、待ち先のシステム（即ち、システムｊ）において、待ち管理テーブルＴ３を参照すれば、デッドロックを検出できる。なお、トランザクションｘの「待ち関係」が解消したとき、トランザクションｘに関する「待ち関係」の情報を待ち先のシステム（即ち、システムｊ）から回収（削除）しないと、いつまでもデッドロックを検出してしまう。そこで、「待ち関係」が解消した場合には、待ち先のシステム（即ち、システムｊ）の待ち管理テーブルＴ３から「待ち関係」を示すグラフを削除・あるいは回収しなければならない。デッドロック検出部（ＤＤ）１５は、このようなグラフ削除・回収機能をも有する。
【００７９】
ウェイトフォーグラフのループが検出されない場合、ウェエイトフォーグラフの先端が他のシステムのトランザクションであれば、当該他のシステムにデッドロックの可能性があることになる。そこで、当該他のシステムにグラフ登録を要求する。他のシステムからのグラフ登録の要求ををを受け付けた場合は、デッドロック検出部（ＤＤ）１５は必要なグラフを登録し、ループ検出を行う。
【００８０】
この待ち関係は、自システム内で生じるときがある。例えば、システムｉで発生したトランザクションｘ1＝Ｔ（ｉ，ｘ1）が自システム（即ち、システムｉ）で発生したトランザクションｙ1＝Ｔ（ｉ，ｙ1）にいて待つとき、自システム（即ち、システムｉ）の待ち管理テーブルＴ３に、

Ｔ（ｉ，ｘ1）→Ｔ（ｉ，ｙ1）
が登録される。ここで、自システム（即ち、システムｉ）の待ち管理テーブルＴ３に、
Ｔ（ｉ，ｙ1）→Ｔ（ｉ，ｘ1）
が登録されているなら、
Ｔ（ｉ，ｘ1）→Ｔ（ｉ，ｙ1）→Ｔ（ｉ，ｙ1）→Ｔ（ｉ，ｘ1）
というループが形成されるので、デッドロックが検出できる。

【００８１】

ところで、分散処理システムにあっては、あるシステムで発生したトランザクションに関係するウェイトフォーグラフ等の登録内容を、そのシステムのローカルウェイトフォーグラフという。また、分散処理システム全体での待ち関係を表現したグラフ，即ち、その分散システムにおける全ローカルウェイトフォーグラフの集合を、グローバルウェイトフォーグラフという。図５は、ローカルウェイトフォーグラフとグローバルウェイトフォーグラフの関係の例を示したものである。

【００８２】

各コンピュータシステムでは、ローカルウェイトフォーグラフのみを待ち管理テーブルＴ３で管理する。ここでは、前記したように、デッドロック検出部（ＤＤ）１５は、自コンピュータシステム内のトランザクション間の待ち関係を表すウェイトフォーグラフ等の登録内容を他のコンピュータシステムに送信しない。一方、デッドロック検出部（ＤＤ）１５は、他のコンピュータシステム内のトランザクションとの間の待ち関係を表すウェイトフォーグラフを、関係を持った他システムにのみ送信する。統計的に見てデッドロックの９０％以上が２つのタスク間で発生することを考慮すると、他システムのトランザクションに関連するデッドロックであっても、ほとんど１回の通信で検出することができる。しかも、自コンピュータシステム内でのデッドロックであれば、通信なしで検出できる。従って、デッドロック検出のための通信のオーバーヘッドを削減できる。

【００８３】

＜待ち時間監視部（ＷＴ）＞

次に、待ち時間監視部（ＷＴ）１３は、待ち管理テーブルＴ３を監視するタイマーである。このタイマーは、待ち管理テーブルＴ３に登録されている「待ち関係」を監視する。そして、その「待ち関係」が登録されてから一定時間経過した時点で、なおその「待ち関係」が継続しているならば、その「待ち関係」にある待ち元のトランザクションが資源獲得要求を再発行するように、リトライ通知を発行する。このリトライ通知は、要求キュー受付部（ＱＲ）１４に投入される。このリトライ通知が要求キュー受付部（ＱＲ）１４に投入されると、デッドロック検出部（ＤＤ）１５は、トランザクション管理部（ＴＭ）１０にリトライ通知を送る。トランザクション管理部（ＴＭ）１０は、このリトライ信号を受けて、待ち関係にあるトランザクションに対し、再度資源獲得要求を出す。

【００８４】

システム間の通信メッセージの遅延・消失やコンピュータシステムの非同期ダウンが発生すると、実際はデッドロック状態であるのにこれを検出できない場合がある。待ち時間監視部（ＷＴ）１３は、このような不都合を防止する。すなわち、分散処理システムにおいて各コンピュータシステム間で通信するとき、通信の欠落によりデッドロック状態を表示するグラフが欠落することがある。すると、デッドロック検出ができなくなる。これを防止するために、待ち関係にあるトランザクションを監視する機構として、前記待ち時間監視部（ＷＴ：Ｗａｔｃｈｄｏｇ　Ｔｉｍｅｒ）１３を設けたのである。

【００８５】

このタイマーは、一定時間以上ウェイトフォーグラフの待ち先の関係にあるトランザクションに対し、前述した様に、デッドロック検出部（ＤＤ）１５を介して、再度資源獲得要求することを促す。すると、トランザクション監視部（ＴＭ：Ｔｒａｎｓａｃｔｉｏｎ　Ｍａｎａｇｅｒ）１０から、再度資源獲得要求が出される。このとき、既に「待ち」が解消されているなら、この資源獲得要求は満たされる。これに対して、待ちが解消されていないなら、他のコンピュータシステムに対して再度ウェイトフォーグラフが送信される。

これによりウェイトフォーグラフ欠落が補われ、デッドロックが検出できる。

【００８６】

＜各部の動作例＞

以下、前記各部の動作をフローチャート図に従って説明する。

〔トランザクション管理部（ＴＭ）の動作〕

図６に示したフローチャートのように、トランザクション管理部（ＴＭ）１０は、ｓｔａｒｔ（開始要求），ａｂｏｒｔ（異常終了），ｃｏｍｍｉｔ（正常終了），資源獲得要求，デッドロック通知，リトライ通知などの各種要求を待つ（ステップＳ１０１）。なお、ここで言うａｂｏｒｔ（異常終了），ｃｏｍｍｉｔ（正常終了）には、他のコンピュータシステムから通知されたものも含む。何れかの要求を受け付ける（ステップＳ１０２）と、トランザクション管理部（ＴＭ）１０は、その要求の種類に従って処理を振り分ける。

【００８７】

ステップＳ１０２で受け付けた要求がｓｔａｒｔ（開始要求）の場合、トランザクション管理部（ＴＭ）１０自身にそのトランザクション（ここでは、仮にＴ（ｉ，ｘ）とする。）を登録し（ステップＳ１０３）、その後の要求を待つ。

【００８８】

ステップＳ１０２で受け付けた要求がａｂｏｒｔ（異常終了）又はｃｏｍｍｉｔ（正常終了）である場合、先ず、その終了するトランザクション（ここでは、仮にＴ（ｉ，ｘ）とする。）を削除する（ステップＳ１０４）。次に、資源管理部（ＲＭ）１１に対し、資源解放要求を発行する（ステップＳ１０５）。その資源開放要求に対する応答を資源管理部（ＲＭ）１１から受けると（ステップＳ１０６）、デッドロック検出部（ＤＤ）１５にグラフ削除要求を出す（ステップＳ１０７）。その後、その要求が自コンピュータシステムからの要求か否かを判定する（ステップＳ１０８）。他コンピュータシステムからの要求であればそのままとする。これに対して、自コンピュータシステムからの要求であれば、他のコンピュータシステムに、ｃｏｍｍｉｔ又はａｂｏｒｔを通知する（ステップＳ１０９）。通知を受けた他のコンピュータシステムでは、ステップＳ１０４乃至１０７の処理を行う。

【００８９】

ステップＳ１０２で受け付けた要求が資源獲得要求である場合、資源管理部（ＲＭ）１１に資源獲得要求を出す（ステップＳ１１０）。その要求に対する応答を資源管理部（ＲＭ）１１から受けたら（ステップＳ１１１）、トランザクションに応答を返す（ステップＳ１１２）。

【００９０】

ステップＳ１０２で受け付けた要求がデッドロック通知である場合、まず、デッドロックとなっているトランザクションの中からａｂｏｒｔさせるべきトランザクションを選択する（ステップＳ１２０）。即ち、デッドロック通知には、デッドロックの関係にある全トランザクション（ここでは、仮にＴ（ｉ，ｘ），Ｔ（ｊ，ｙ）とする。）の特定が含まれている。トランザクション管理部（ＴＭ）１０は、このデッドロック通知に含まれているトランザクション名からａｂｏｒｔさせるべきトランザクションを選択するのである。従って、トランザクション管理部（ＴＭ）１０は、他のコンピュータシステムのトランザクションをも、ａｂｏｒｔ対象として特定することができる。次いで、トランザクション管理部（ＴＭ）１０は、選択されたトランザクションにａｂｏｒｔすべき旨の通知をする（ステップＳ１２１）。選択されたトランザクションが他のコンピュータシステムのものである場合には、当該他のコンピュータシステムのトランザクション管理部（ＴＭ）１０を介して、選択されたトランザクションにａｂｏｒｔすべき旨を通知する。

【００９１】

ステップＳ１０２で受け付けた要求がリトライ通知である場合、まず、資源管理部（ＲＭ）１１に資源獲得要求を出す（ステップＳ１３０）。その要求に対する応答を資源管理部（ＲＭ）１１からもらったら（ステップＳ１３１）、トランザクションに応答を返す（ステップＳ１３１）。

【００９２】
〔資源管理部（ＲＭ）の動作〕
図７に示したフローチャートのように、資源管理部（ＲＭ）１１は、まず、資源獲得要求及び資源解放要求を待つ（ステップＳ２０１）。何れかの要求があり、それが受理されると（ステップＳ２０２）、テーブル上に示された資源をロックしようとし、その関係を資源獲得テーブルＴ１に登録する（ステップＳ２０３）。即ち、どのトランザクションがどの資源をロックしようとするのかを登録する。
【００９３】
その後、要求が資源獲得要求か資源解放要求かを判定する（ステップＳ２０４）。要求が資源獲得要求の場合、ロック管理部（ＬＭ）１２にロック獲得要求を出す（ステップＳ２０５）。これに対して、要求が資源解放要求の場合、ロック管理部（ＬＭ）１２にロック解放要求を出す（ステップＳ２０６）。
【００９４】
そして、ロック獲得要求又はロック解放要求に対する応答（ｏｋ／ｎｏ）をロック管理部（ＬＭ）１２から受けた後（ステップＳ２０７）、トランザクション管理部（ＴＭ）１０に応答（ｏｋ／ｎｏ）を返す（ステップＳ２０８）。
【００９５】
〔ロック管理部（ＬＭ）の動作〕
図８に示したフローチャートのように、資源管理部（ＲＭ）１１におけるステップＳ２０５又はステップＳ２０６の要求があると（ステップＳ３０１）、ロック管理部（ＬＭ）１２は、要求を受け付ける（ステップＳ３０２）。その後、要求がロック獲得要求かロック解放要求かを判定する（ステップＳ３０３）。要求がロック獲得要求である場合には、ロック管理部（ＬＭ）１２は、ロック獲得が可能か否かを判定する（ステップＳ３０４）。
【００９６】
資源のロックが可能であれば、そのロック状態をロック管理テーブルＴ２に登録する（ステップＳ３０５）。資源のロックが不可能であれば、「待ち関係」であるので、要求側のトランザクションと待ち先のトランザクションとの関係をウェイトフォーグラフとして待ち管理テーブルＴ３に登録する旨を、デッドロック検出部（ＤＤ）１５に対して要求する（ステップＳ３０６）。その後、資源管理部（ＲＭ）１１にロックできなかった旨（ｎｏ）を返答する（ステップＳ３０９）。
【００９７】
ステップＳ３０３において要求が資源解放要求であると判定された場合、ロック管理テーブルＴ２からロックの登録を削除する（ステップＳ３０７）。ロックの登録（ステップＳ３０５）とその削除（ステップＳ３０７）の後は、その完了（ｏｋ）を示す応答を、資源管理部（ＲＭ）１１に返す（ステップＳ３０８）。
【００９８】
〔デッドロック検出部（ＤＤ）の動作〕
図９に示したフローチャート図のように、デッドロック検出部（ＤＤ）１５には、グラフ登録要求，グラフ削除要求，及びリトライ通知が、要求キュー受付部（ＱＲ）１４に受け付けられる。従って、その要求があると（ステップＳ４０１）、要求キュー受付部（ＱＲ）１４から要求を取り出し（ステップＳ４０２）、要求の種別を判定する（ステップＳ４０３）。
【００９９】
要求が、グラフ登録である場合には、先ず、待ち管理テーブルＴ３にウェイトフォーグラフを登録する（ステップＳ４０４）。但し、待ち管理テーブルＴ３を検索した結果同一グラフが既に登録されていれば、そのグラフは登録しない。次いで、登録したグラフにつき、グラフの先端までたどる（ステップＳ４０５）。たどった結果によって、ループが形成されているか判断する（ステップＳ４０６）。ループが形成されていれば、トランザクション管理部（ＴＭ）１０にデッドロックを通知する（ステップＳ４０７）。ループが形成されていなければ、グラフの先端が自コンピュータシステムか否かを判定する（ステップ

Ｓ４０８）。自コンピュータシステムであればそのままステップＳ４０１に戻る。これに対して、他コンピュータシステムであれば、その他コンピュータシステムのデッドロック検出部（ＤＤ）１５に当該ウェイトフォーグラフを送信して、その他システムの待ち管理テーブルＴ３に当該グラフを登録させる（ステップＳ４０９）。

【０１００】

次に、ステップＳ４０３において、要求がグラフの削除であるときは、待ち管理テーブルＴ３を検索して、該当するグラフを探す（ステップＳ４１０）。該当グラフを探しあてたら、該当グラフを削除する（ステップＳ４１１）。その後、待ち関係が解除されたトランザクションを動作させるため、トランザクション管理部（ＴＭ）１０にリトライ通知をする（ステップＳ４１２）。

【０１０１】

ステップＳ４０３で要求がリトライ通知であるとき、まず、リトライするトランザクションのウェイトフォーグラフを削除する（ステップＳ４２０）。次いで、トランザクション管理部（ＴＭ）１０にリトライ通知をする（ステップＳ４２１）。

【０１０２】

〔待ち時間監視部（ＷＴ）の動作〕

図１０に示したフローチャートのように、待ち時間監視部（ＷＴ）１３は、待ち管理テーブルＴ３に登録されている各トランザクション（Ｔ（ｉ，ｘ）等）を順次検索する（ステップＳ５０１）。次いで、そのトランザクションｘが「待ち関係」にあるか否かを判断する（ステップＳ５０２）。検索されたトランザクションが「待ち関係」でなければ、ステップＳ５０１に戻り、次のトランザクションを検索する。

【０１０３】

これに対して、検索されたトランザクションが「待ち関係」であれば、タイムカウントを開始し、タイムアウトとなったら（ステップＳ５０３）、デッドロック検出部（ＤＤ）１５にリトライ通知を行う（ステップＳ５０４）。ステップＳ５０３でタイムアウトになる前に「待ち関係」が解消されたなら、ステップＳ５０１に戻る（ステップＳ５０２）。

【０１０４】

＜具体的なデッドロック検出の例＞

次に、以上の構成におけるデッドロック検出例を、３通りの場合に沿って説明する。

［例１　自コンピュータシステム内におけるデッドロック検出］

例１は、自コンピュータシステム内におけるデッドロック検出の例で、具体的には以下の動作を行う。ここでは、他のコンピュータシステムとの間に通信が発生しないことが解る。

【０１０５】

(１)　先ず、トランザクションＴ（１，１）がトランザクション管理部（ＴＭ）１０に対してトランザクションの開始を通知したとする。

(２)　すると、トランザクション管理部（ＴＭ）１０はデッドロック検出部（ＤＤ）１５にトランザクションＴ（１，１）の登録を要求する。

(３)　一方、トランザクションＴ（１，２）がトランザクション管理部（ＴＭ）１０に対してトランザクションの開始を通知したとする。

【０１０６】

(４)　すると、トランザクション管理部（ＴＭ）１０はデッドロック検出部（ＤＤ）１５にトランザクションＴ（１，２）の登録を要求する。

(５)　いま、トランザクションＴ（１，１）がトランザクション管理部（ＴＭ）１０に資源Ａを要求したとする。

(６)　すると、トランザクション管理部（ＴＭ）１０が資源管理部（ＲＭ）１１に資源Ａの獲得を要求する。

【０１０７】

(７)　すると、資源管理部（ＲＭ）１１がロック管理部（ＬＭ）１２に資源Ａのロック獲得を要求する。

(８) 資源Ａが未ロックであれば、ロック管理部（ＬＭ）１２が資源管理部（ＲＭ）１１にＯＫを応答する。
(９) すると、資源管理部（ＲＭ）１１がトランザクション管理部（ＴＭ）１０にＯＫを応答する。
【０１０８】
(10) 一方、トランザクションＴ（１，２）がトランザクション管理部（ＴＭ）１０に資源Ｂを要求したとする。
(11) すると、トランザクション管理部（ＴＭ）１０が資源管理部（ＲＭ）１１に資源Ｂを獲得を要求する。
(12) すると、資源管理部（ＲＭ）１１がロック管理部（ＬＭ）１２に資源Ｂのロック獲得を要求する。
【０１０９】
(13) 資源Ｂが未ロックであれば、ロック管理部（ＬＭ）１２が資源管理部（ＲＭ）１１にＯＫを応答する。
(14) すると、資源管理部（ＲＭ）１１がトランザクション管理部（ＴＭ）１０にＯＫを応答する。
(15) この状態において、トランザクションＴ（１，１）がトランザクション管理部（ＴＭ）１０に資源Ｂを要求したとする。
【０１１０】
(16) すると、トランザクション管理部（ＴＭ）１０が資源管理部（ＲＭ）１１に資源Ｂの獲得を要求する。
(17) すると、資源管理部（ＲＭ）１１がロック管理部（ＬＭ）１２に資源Ｂのロック獲得を要求する。
【０１１１】
(18) ところが、資源Ｂはトランザクションｔ（１，２）によって既にロック済みであるので、Ｔ（１，１）がトランザクションＴ（１，２）に対して待つことになる。そこで、ロック管理部（ＬＭ）１２がデッドロック検出部（ＤＤ）１５にグラフ
Ｔ（１，１）→Ｔ（１，２）
の登録を要求する。要求を受けたデッドロック検出部（ＤＤ）１５は、このグラフを待ち管理テーブルＴ３に登録する。
【０１１２】
(19) 一方、トランザクションＴ（１，２）がトランザクション管理部（ＴＭ）１０に資源Ａを要求したとする。
(20) すると、トランザクション管理部（ＴＭ）１０が資源管理部（ＲＭ）１１に資源Ａの獲得を要求する。
(21) すると、資源管理部（ＲＭ）１１がロック管理部（ＬＭ）１２に資源Ａのロック獲得を要求する。
【０１１３】
(22) ところが、資源Ａはトランザクションｔ（１，１）によって既にロック済みであるので、Ｔ（１，２）がトランザクションＴ（１，１）に対して待つことになる。そこで、ロック管理部（ＬＭ）１２がデッドロック検出部（ＤＤ）１５にグラフ
Ｔ（１，２）→Ｔ（１，１）
の登録を要求する。要求を受けたデッドロック検出部（ＤＤ）１５は、このグラフを待ち管理テーブルＴ３に登録する。
(23) デッドロック検出部（ＤＤ）１５がループを検出し、デッドロック発生をトランザクション管理部（ＴＭ）１０に通知する。
【０１１４】
［例２　２つのコンピュータシステム間における２つのトランザクションのデッドロック検出］
例２は、２つのコンピュータシステム（システム１，システム２）間でデッドロックが発

生する場合を示している。ここでは、デッドロック検出のための通信が１回で済むことがわかる。
(１) 先ず、システム１におけるトランザクションＴ（１，１）が、システム１のトランザクション管理部（ＴＭ）１０に対しトランザクションの開始を通　知したとする。
(２) すると、システム１のトランザクション管理部（ＴＭ）１０は、デッドロック検出部（ＤＤ）１５にトランザクションＴ（１，１）の登録を要求する。
(３) いま、トランザクションＴ（１，１）が、システム１のトランザクション管理部（ＴＭ）１０に、資源Ａを要求したとする。
【０１１５】
(４) すると、システム１のトランザクション管理部（ＴＭ）１０が、資源管理部（ＲＭ）１１に資源Ａの獲得を要求する。
(５) すると、システム１の資源管理部（ＲＭ）１１が、ロック管理部（ＬＭ）１２に資源Ａのロック獲得を要求する。
(６) 資源Ａが未ロックであれば、システム１のロック管理部（ＬＭ）１２が、資源管理部（ＲＭ）１１にＯＫを応答する。
【０１１６】
(７) すると、システム１の資源管理部（ＲＭ）１１が、トランザクション管理部（ＴＭ）１０にＯＫを応答する。
(１)’一方、システム２におけるトランザクショントランザクションＴ（２，１）が、システム２のトランザクション管理部（ＴＭ）１０に対しトランザクションの開始を通知したとする。
(２)’すると、システム２のトランザクション管理部（ＴＭ）１０は、デッドロック検出部（ＤＤ）１５にトランザクションＴ（２，１）の登録を要求する。
【０１１７】
(３)’いま、トランザクションＴ（２，１）が、システム２のトランザクション管理部（ＴＭ）１０に、資源Ｂを要求したとする。
(４)’すると、システム２のトランザクション管理部（ＴＭ）１０が、資源管理部（ＲＭ）１１に資源Ｂの獲得を要求する。
(５)’すると、システム２の資源管理部（ＲＭ）１１が、ロック管理部（ＬＭ）１２に資源Ｂのロック獲得を要求する。
(６)’資源Ｂが未ロックであれば、システム２のロック管理部（ＬＭ）１２が、資源管理部（ＲＭ）１１にＯＫを応答する。
【０１１８】
(７)’すると、システム２の資源管理部（ＲＭ）１１が、トランザクション管理部（ＴＭ）１０にＯＫを応答する。
(８) 以上の状況下において、トランザクションＴ（１，１）がシステム１のトランザクション管理部（ＴＭ）１０に資源Ｂを要求したとする。
(９) すると、システム１のトランザクション管理部（ＴＭ）１０が、資源管理部（ＲＭ）１１に資源Ｂの獲得を要求する。
【０１１９】
(10) すると、システム１の資源管理部（ＲＭ）１１が、ロック管理部（ＬＭ）１２に資源Ｂのロック獲得を要求する。
(11) ところが、資源Ｂはシステム２のトランザクションＴ（２，１）によって既にロック済みであるので、トランザクションＴ（１，１）がトランザクションＴ（２，１）に対して待つことになる。そこで、システム１のロック管理部（ＬＭ）１２が、デッドロック検出部（ＤＤ）１５にグラフ
Ｔ（１，１）→Ｔ（２，１）
の登録を要求する。
【０１２０】
(12) この要求を受けて、システム１のデッドロック検出部（ＤＤ）１５は、グラフＴ（

１，１）→Ｔ（２，１）を待ち管理テーブルＴ３に登録し、待ち管理テーブルＴ３に登録されたグラフにループが形成されているかどうか（デッドロックが発生しているかどうか）を判断し、デッドロックを検出しないときにシステム２にグラフＴ（１，１）→Ｔ（２，１）を送信する。(13) システム２のデッドロック検出部（ＤＤ）１５が、このグラフＴ（１，１）→（２，１）を受信し、これをシステム２の待ち管理テーブルＴ３に登録する。

【０１２１】

(14) この後で、トランザクションＴ（２，１）がシステム２のトランザクション管理部（ＴＭ）１０に資源Ａを要求したとする。

(15) すると、システム２のトランザクション管理部（ＴＭ）１０が、資源管理部（ＲＭ）１１に資源Ａの獲得を要求する。

(16) すると、システム２の資源管理部（ＲＭ）１１が、ロック管理部（ＬＭ）１２に資源Ａのロック獲得を要求する。

【０１２２】

(17) ところが、資源Ａはシステム１のトランザクションＴ（１，１）によって既にロック済みであるので、トランザクションＴ（２，１）がトランザクションＴ（１，１）に対して待つことになる。そこで、システム２のロック管理部（ＬＭ）１２が、デッドロック検出部（ＤＤ）１５にグラフＴ（２，１）→Ｔ（１，１）の登録を要求する。

(18) システム２のデッドロック検出部（ＤＤ）１５は、グラフＴ（２，１）→Ｔ（１，１）を待ち管理テーブルＴ３に登録し、待ち管理テーブルＴ３に登録されたグラフにループが形成されているか判断する。その結果、システム２のデッドロック検出部（ＤＤ）１５は、ループを検出し、デッドロック発生をシステム２のトランザクション管理部（ＴＭ）１０に通知する。

【０１２３】

［例３　２つのコンピュータシステム間における２つのトランザクションのデッドロック検出中に、メッセージの消失発生］

例３では、２つのコンピュータシステム（システム１，システム２）間での通信エラーにより、メッセージ消失が発生した場合の例である。

(１) 先ず、システム１におけるトランザクショントランザクションＴ（１，１）が、システム１のトランザクション管理部（ＴＭ）１０に対しトランザクションの開始を通知したとする。

【０１２４】

(２) すると、システム１のトランザクション管理部（ＴＭ）１０は、デッドロック検出部（ＤＤ）１５にトランザクションＴ（１，１）の登録を要求する。

(３) いま、トランザクションＴ（１，１）が、システム１のトランザクション管理部（ＴＭ）１０に、資源Ａを要求したとする。

(４) すると、システム１のトランザクション管理部（ＴＭ）１０が、資源管理部（ＲＭ）１１に資源Ａの獲得を要求する。

(５) すると、システム１の資源管理部（ＲＭ）１１が、ロック管理部（ＬＭ）１２に資源Ａのロック獲得を要求する。

【０１２５】

(６) 資源Ａが未ロックであれば、システム１のロック管理部（ＬＭ）１２が、資源管理部（ＲＭ）１１にＯＫを応答する。

(７) すると、システム１の資源管理部（ＲＭ）１１が、トランザクション管理部（ＴＭ）１０にＯＫを応答する。

(１)’一方、システム２におけるトランザクションＴ（２，１）が、システム２のトランザクション管理部（ＴＭ）１０に対しトランザクションの開始を通知したとする。

【０１２６】

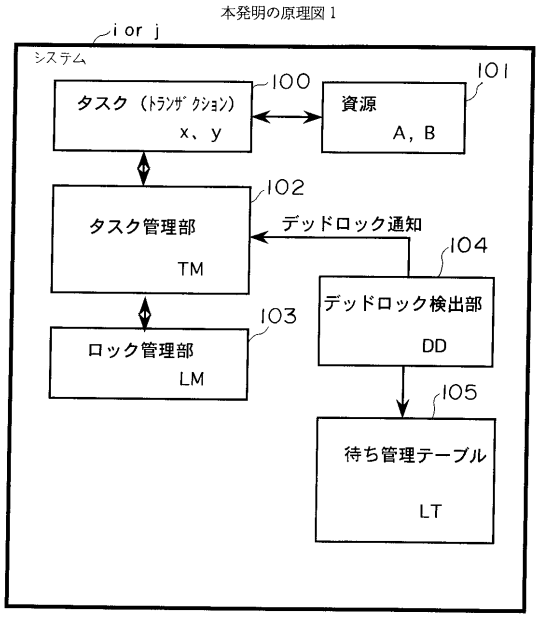
(２)’すると、システム２のトランザクション管理部（ＴＭ）１０は、デッドロック検出部（ＤＤ）１５にトランザクションＴ（２，１）の登録を要求する。

(３)’いま、トランザクションＴ（２，１）が、システム２のトランザクション管理部（ＴＭ）１０に、資源Ｂを要求したとする。
(４)’すると、システム２のトランザクション管理部（ＴＭ）１０が、資源管理部（ＲＭ）１１に資源Ｂの獲得を要求する。
【０１２７】
(５)’すると、システム２の資源管理部（ＲＭ）１１が、ロック管理部（ＬＭ）１２に資源Ｂのロック獲得を要求する。
(６)’資源Ｂが未ロックであれば、システム２のロック管理部（ＬＭ）１２が、資源管理部（ＲＭ）１１にＯＫを応答する。
【０１２８】
(７)’すると、システム２の資源管理部（ＲＭ）１１が、トランザクション管理部（ＴＭ）１０にＯＫを応答する。
(８) 以上の状況下において、トランザクションＴ（１，１）がシステム１のトランザクション管理部（ＴＭ）１０に資源Ｂを要求したとする。
(９) すると、システム１のトランザクション管理部（ＴＭ）１０が、資源管理部（ＲＭ）１１に資源Ｂの獲得を要求する。
【０１２９】
(10) すると、システム１の資源管理部（ＲＭ）１１が、ロック管理部（ＬＭ）１２に資源Ｂのロック獲得を要求する。
(11) ところが、資源Ｂはシステム２のトランザクションＴ（２，１）によって既にロック済みであるので、トランザクションＴ（１，１）がトランザクションＴ（２，１）に対して待つことになる。そこで、システム１のロック管理部（ＬＭ）１２が、デッドロック検出部（ＤＤ）１５にグラフ
Ｔ（１，１）→Ｔ（２，１）
の登録を要求する。
【０１３０】
(12) この要求を受けて、システム１のデッドロック検出部（ＤＤ）１５は、このグラフをシステム１の待ち管理テーブルＴ３に登録する。これと同時に、システム１のデッドロック検出部（ＤＤ）１５は、システム２にグラフ
Ｔ（１，１）→Ｔ（２，１）
を送信する。
(13) ただし、その送信内容は、通信エラーにより消失して、システム２に届かなかった。
【０１３１】
(14) この後で、トランザクションＴ（２，１）がシステム２のトランザクション管理部（ＴＭ）１０に資源Ａを要求したとする。
(15) すると、システム２のトランザクション管理部（ＴＭ）１０が、資源管理部（ＲＭ）１１に資源Ａの獲得を要求する。
(16) すると、システム２の資源管理部（ＲＭ）１１が、ロック管理部（ＬＭ）１２に資源Ａのロック獲得を要求する。
【０１３２】
(17) ところが、資源Ａはシステム１のトランザクションＴ（１，１）によって既にロック済みであるので、トランザクションＴ（２，１）がトランザクションＴ（１，１）に対して待つことになる。そこで、システム２のロック管理部（ＬＭ）１２が、デッドロック検出部（ＤＤ）１５にグラフ
Ｔ（２，１）→Ｔ（１，１）
の登録を要求する。この時点で、実際にはデッドロック状態が生じている。
【０１３３】
しかしながら、メッセージ消失によりデッドロック状態は検出できないので、デッドロック状態が持続することになる。

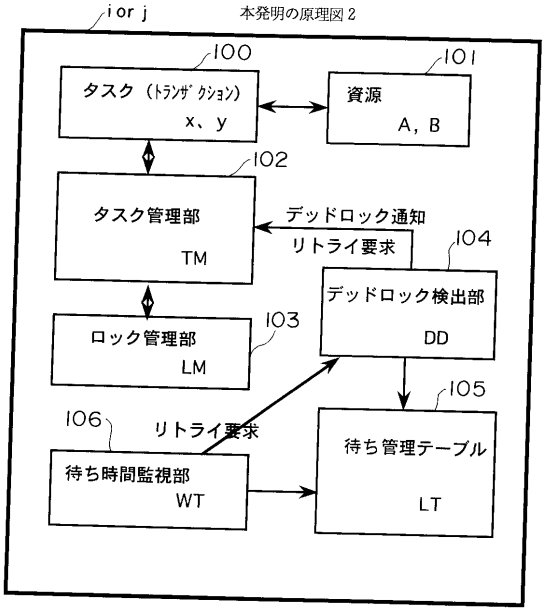
(18) 一定時間後、システム１の待ち時間監視部（ＷＴ）１３が起動され、システム１のデッドロック検出部（ＤＤ）１５に対してリトライ通知を行う。
(19) すると、システム１のデッドロック検出部（ＤＤ）１５は、トランザクション管理部（ＴＭ）１０にトランザクションＴ（１，１）のリトライを通知する。
【０１３４】
(20) リトライ通知に従って、システム１のトランザクション管理部（ＴＭ）１０が、資源管理部（ＲＭ）１１に資源Ｂの獲得を要求する。
(21) すると、システム１の資源管理部（ＲＭ）１１が、ロック管理部（ＬＭ）１２に資源Ｂのロック獲得を再度要求する。
(22) ところが、資源Ｂはシステム２のトランザクションＴ（２，１）によって既にロック済みであるので、トランザクションＴ（１，１）がトランザクションＴ（２，１）に対して待つことになる。そこで、システム１のロック管理部（ＬＭ）１２が、デッドロック検出部（ＤＤ）１５にグラフ
Ｔ（１，１）→Ｔ（２，１）
の登録を再度要求する。
【０１３５】
(23) この要求を受けて、システム１のデッドロック検出部（ＤＤ）１５は、このグラフをシステム１のウェイトフォーグラフテーブルＴ３に登録する。これと同時に、システム２にグラフ
Ｔ（１，１）→Ｔ（２，１）
を再度送信する。
(24) システム２のデッドロック検出部（ＤＤ）１５が、このグラフ
Ｔ（１，１）→Ｔ（２，１）
を受信し、これをシステム２の待ち管理テーブルＴ３に登録する。これにより、グラフの欠落が補われる。
(25) システム２のデッドロック検出部（ＤＤ）１５がループを検出し、デッドロック発生をシステム２のＴＭに通知する。
【０１３６】
【発明の効果】
本発明では、以上説明したように、タスク（トランザクション）による資源のロック状態を管理するロック管理部（ＬＭ）１０３と、デッドロック検出部（ＤＤ）１０４とを分離し、双方が非同期に動作するようにした。そのため、タスク（トランザクション）が新たに発生して資源を要求しても、待ちが発生せずロックが獲得できる場合は、デッドロック検出部（ＤＤ）１０４を介せずに動作できる。従って、システムの円滑な運用が図れ、処理の高速化を図れる。また、ロックが獲得できない場合でも、ロック状態（グラフ）の登録やデッドロックの検出はロックの要求とは非同期に動作するので影響は小さい。
【０１３７】
特に、デッドロックを少なくするように設計されたシステムで、デッドロック検出の与える影響は極めて小さくなる。
本発明が分散システムに適用された場合、システムは他のシステムと待ち関係に陥った場合にのみデッドロックのための通信を行う。したがって、自システム内のみのデッドロックの検出では通信は発生しない。他システムとの関連があった場合も、デッドロックの９０％以上が２者間で発生することから、ほとんどの場合、１回の通信でデッドロックは検出される。このため、通信のオーバーヘッドを削減でき、効率のよいシステム運用を図ることができる。
【０１３８】
また、本発明で、待ち時間監視部（ＷＴ）１３を設けた場合、分散システムでのメッセージ通信中にメッセージが遅延した場合や消失した場合でも、待ち時間監視部（ＷＴ）１３が再びデッドロック検出の契機を与えるため、すべてのデッドロックを検出することができる。

【図面の簡単な説明】
【図１】　本発明の原理図１
【図２】　本発明の原理図２
【図３】　デッドロックを示す説明図
【図４】　実施例を示すブロック図
【図５】　ローカルＷＦＧとグローバルＷＦＧの関係を示す図
【図６】　トランザクション管理部の動作を示すフローチャート
【図７】　資源管理部の動作を示すフローチャート
【図８】　ロック管理部の動作を示すフローチャート
【図９】　デッドロック検出部の動作を示すフローチャート
【図１０】　待ち時間監視部の動作を示すフローチャート
【符号の説明】
１０　　トランザクション管理部
１１　　資源管理部
１２　　ロック管理部
１３　　待ち時間監視部
１４　　要求キュー受付部
１５　　デッドロック検出部
２０　　データベース
Ｔ３　　待ち管理テーブル

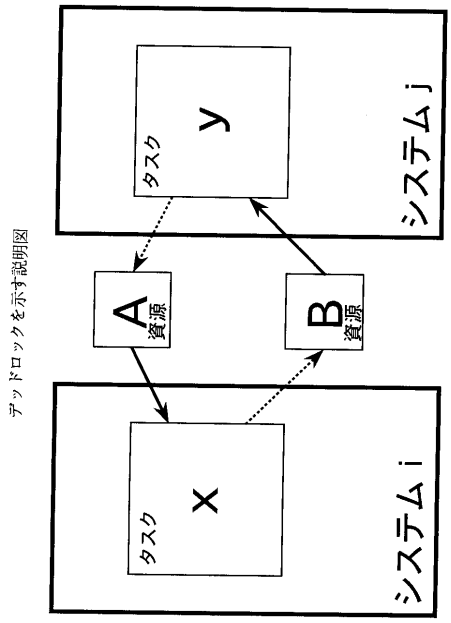
【図１】



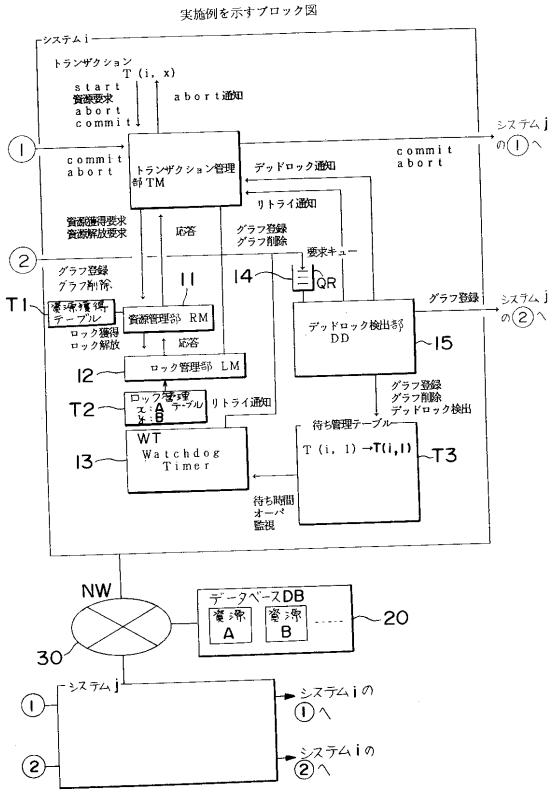
【図２】



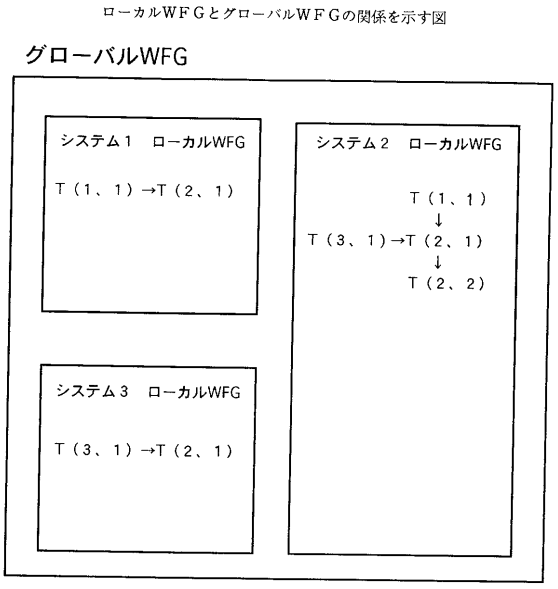
【図３】


デッドロックを示す説明図
システムj
タスク
y
A
資源
B
資源
システムi
タスク
x

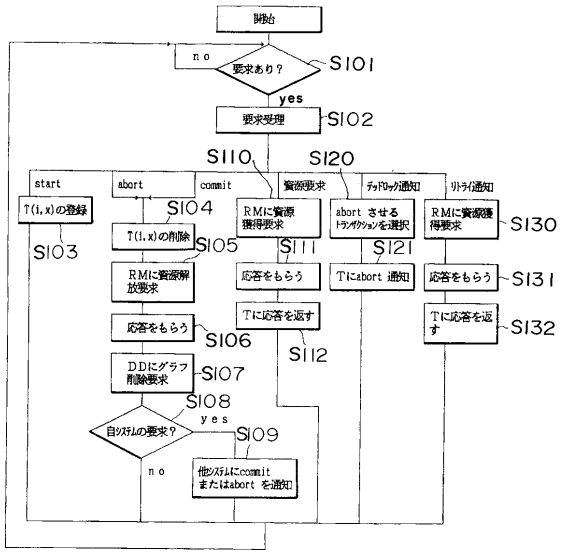
【図４】


実施例を示すブロック図
システムi
トランザクション
T (i, x)
start
資源要求
abort
commit
abort通知
commit
abort
トランザクション管理部 TM
デッドロック通知
commit
abort
システムj
の①へ
リトライ通知
資源獲得要求
資源解放要求
応答
グラフ登録
グラフ削除
要求キュー
QR
11
14
グラフ登録
グラフ削除
T1
資源管理テーブル
資源管理部 RM
ロック獲得
ロック解放
応答
デッドロック検出部 DD
15
グラフ登録
システムj
の②へ
12
ロック管理部 LM
T2
ロック管理テーブル
リトライ通知
グラフ登録
グラフ削除
デッドロック検出
待ち管理テーブル
WT
Watchdog Timer
13
T (i, 1) →T(i,1)
T3
待ち時間オーバ監視
NW
30
データベースDB
資源A
資源B
20
システムj
①
②
システムiの①へ
システムiの②へ

【図５】

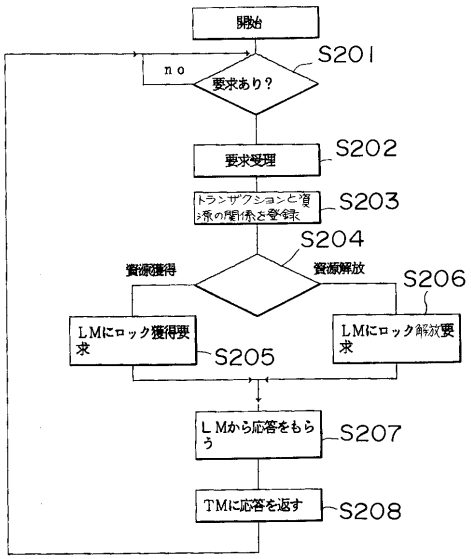

ローカルWFGとグローバルWFGの関係を示す図
グローバルWFG
システム１ ローカルWFG
T（1、1）→T（2、1）
システム２ ローカルWFG
T（1、1）
↓
T（3、1）→T（2、1）
↓
T（2、2）
システム３ ローカルWFG
T（3、1）→T（2、1）

【図６】


トランザクション管理部の動作を示すフローチャート図
開始
no
要求あり？
S101
yes
要求受理
S102
S110
S120
start
abort
commit
資源要求
デッドロック通知
リトライ通知
T(i,x)の登録
S103
S104
RMに資源獲得要求
abort させるトランザクションを選択
RMに資源獲得要求
S130
T(i,x)の削除
S105
S111
S121
RMに資源解放要求
応答をもらう
Tにabort 通知
応答をもらう
S131
Tに応答を返す
Tに応答を返す
S132
応答をもらう
S106
S112
DDにグラフ削除要求
S107
S108
yes
自システムの要求？
S109
no
他システムにcommitまたはabort を通知

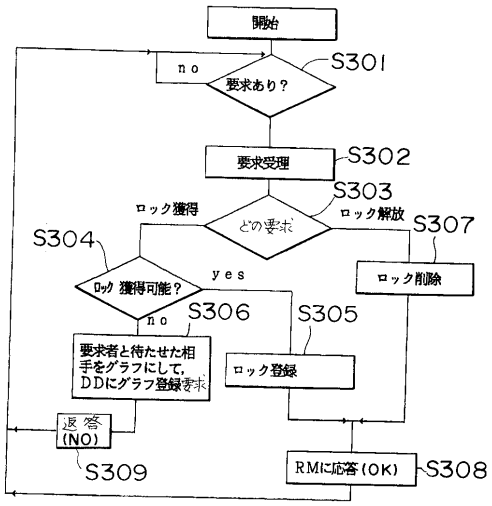
【図７】

資源管理部の動作を示すフローチャート図



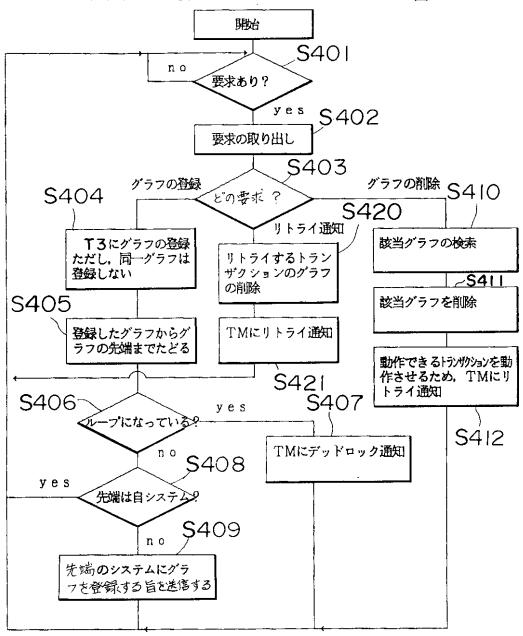
【図８】

ロック管理部の動作を示すフローチャート図



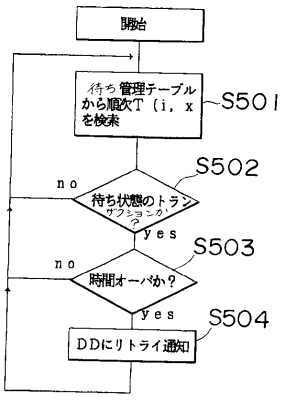
【図９】

デッドロック検出部の動作を示すフローチャート図



【図１０】

待ち時間監視部の動作を示すフローチャート図



フロントページの続き

合議体
審判長　吉岡　浩
審判官　松浦　功
審判官　山中　実

(56)参考文献　特開平２－２４７７７０（ＪＰ，Ａ）

(58)調査した分野(Int.Cl.[7]，ＤＢ名)
G06F9/46
G06F15-16-15/177