分散形システムにおけるデッドロックの検出と回復

5X-5

緒方 正暢[†]　菊野 亨[††]　吉田 典可[††]

[†]日本アイ・ビー・エム株式会社　サイエンス・インスティチュート
[††]広島大学工学部

1．まえがき

分散処理の環境下で資源の共有を行なうシステムにおいては，デッドロックの解決は重要な問題の１つである．最近では，分散形データベースシステムにおけるデッドロックの検出と回復が注目されている．分散処理環境では，複数のコンピュータサイトにまたがるデッドロックが存在する可能性があり，通信遅延時間等の要因を考慮した効率の良い分散形アルゴリズムが望まれる．

本稿では，N-out-of-M要求に基づく分散形システムの一般的なモデル[2],[3]におけるデッドロックの検出と回復を行なう分散形アルゴリズムについて議論する．

2．N-out-of-Mモデル

分散形システムはプロセスの集合であり，各プロセスはメッセージによってのみ通信ができるものとする．N-out-of-M(1≦N≦M)モデルでは，アクティブな（他のプロセスを待っていない）プロセスpは他のM個のプロセスに対して要求を出し，待ち状態になる．その後，少なくともN個のプロセスから承諾が得られるまで，pはその実行を待ち続ける．N個のプロセスから承諾が得られると，残りのM－N個のプロセスに対する要求を取り消してpは再びアクティブになる．N-out-of-Mモデルでは，N＝MのときAND要求，N＝１のときOR要求を表わしている．

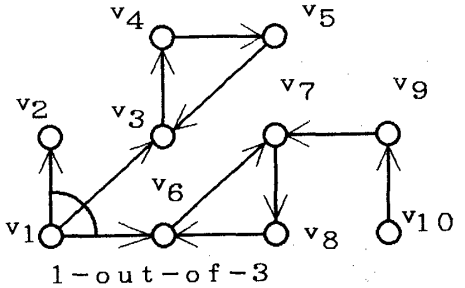

図１．　待ちグラフG＝(V,E)

プロセス間の待ちの関係を待ちグラフと呼ばれる有向グラフG＝(V,E)で表わす．G上の各節点はプロセスを，各有向枝はプロセス間の待ちの関係を表わす．図１に待ちグラフGの一例を示す．同図でv_1は1-out-of-3要求を，他の節点は1-out-of-1要求を出して待ち状態にある．図１の場合，v_2だけがアクティブになっている．

待ちグラフG＝(V,E)上の変換Ｔ１,Ｔ２を次に定義する．節点v∈Vがアクティブになるまでに受け取るべき承諾の数を♯(v)で表わす．

Ｔ１：アクティブな節点v∈VからVに含まれるｒ個の節点に枝を加え，♯(v)＝k (1≦k≦r)とする．

Ｔ２：節点v∈Vからアクティブな節点に向かう枝を１つ削除し，♯(v)の値を１減らす．もし，♯(v)＝０となれば，vの出力枝を全て削除し，vはアクティブになる．

［定義１］　節点v∈Vがアクティブになる変換Ｔ１，Ｔ２の有限系列が存在するなら，節点vはライブであると定める．そうした系列が存在しないとき，節点vはデッドロックであると言う．

［注１］　N-out-of-Mモデルでは，G上の有向閉路はデッドロックが存在するための十分条件ではない．

3．分散形アルゴリズム

ここで提案する分散形アルゴリズムは，任意のプロセスから起動され，デッドロックの検出を行う．起動するプロセスをイニシエータと呼ぶ．もし，デッドロックが検出されると，デッドロックに陥っているプロセスの中からいくつかを選び，その処理を中止させるという回復処理を行う．イニシエータとしては，例えば，ある時間Ｔ以上の間待ち状態にあるプロセスが選ばれる．イニシエータの選び方は，分散形システムに依存するので，本稿では議論しない．

また，待ちグラフGはシステムのスナップショット[1]から構成され，その形状はアルゴリズムの実行中に変化しないものと仮定する．G上の節点uからvへ有向道が存在するときvはuから到達可能であるという．

Deadlock Detection and Resolution in Distributed Systems
Masanobu OGATA[†], Tohru KIKUNO[††], Noriyoshi YOSHIDA[††]
[†]Science Institute, IBM Japan, Ltd., [††]Hiroshima University

3．1　デッドロックの検出

検出アルゴリズムは，イニシエータから到達可能なG上の各節点v∈Vがデッドロックに陥っているかどうかを決定する分散形アルゴリズムである[2]．本アルゴリズムは基本的に2つのフェーズNotifyとActivateから構成されている．Notifyフェーズでは節点vから到達可能な節点を全て探索し，アクティブな節点を見つける．ActivateフェーズではNotifyフェーズで見つかったアクティブ節点から開始して変換T2をシミュレートし，ライブな節点を決定する．2つのフェーズは4種類のメッセージを用いて並列的に実行される．

本アルゴリズムは全てのライブ節点を見つけた後停止する．その時，イニシエータpはpから到達可能な節点の集合REACH，ライブ節点の集合LIVE(⊂REACH)を得る．定義1より，REACH≠LIVEであればデッドロックに陥った節点の集合D(=REACH-LIVE)が検出されたことになる．

［定理1］検出アルゴリズムの計算量はメッセージ数4･|E|，時間2d+1[hop]である．ただし，時間計算量の評価に当たってはシステムが同期して動作し，しかも，隣接プロセス間での各メッセージの転送がある単位時間(hop)で行われるものと仮定する．

図2に，v_1をイニシエータとした時のデッドロックの検出の例を示す．アルゴリズムが停止した時，REACH=$\{v_1,v_2,v_3,v_4,v_5,v_6,v_7,v_8\}$，LIVE=$\{v_1,v_2\}$である．同図中ではライブ節点を●で示している．

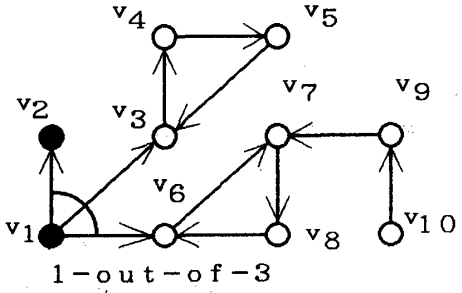

図2．デッドロックの検出の例

3．2　デッドロックの回復

節点の集合A(A⊂D)に対し，Aに含まれる節点を中止したときにデッドロックから回復される節点をセーフと呼び，その集合をSAFE(A)で表す．SAFE(A)=Dであるとき，Aを許容解と呼ぶ．実用上の立場から，|A|はなるべく少なく抑えることが望まれる．ところが，|A|が最小となる集合Aを求める最適化問題はNP困難となる[3]．ここでは，許容解を求める1つのヒューリスティック解法を新たに提案する．

回復アルゴリズムは検出アルゴリズムのイニシエータから起動される分散形アルゴリズムである[3]．本アルゴリズムは2つのフェーズVisitとSafeから構成され，それらが交互に実行される．Visitフェーズではグラフ上を深さ優先探索法に基づきメッセージを送りG上の有向閉路に含まれ，かつ，Dに属する節点を見つけ，それを中止節点に選ぶ．Safeフェーズでは選ばれた中止節点によってセーフになる節点を探索し，セーフになった節点をDから除く．Dが空になるまでこれらのフェーズを繰返す．ただし，Visitフェーズでの探索法における枝の選択の順番はランダムに決めるものとする．

［定理2］回復アルゴリズムの計算量はメッセージ数4･|E|，時間2(|E|+|V|･d)[hop]である．

図3にデッドロックの回復の例を示す．この例では中止節点の集合Aとしてv_3とv_6が選ばれる．同図中では中止節点を◎で示している．

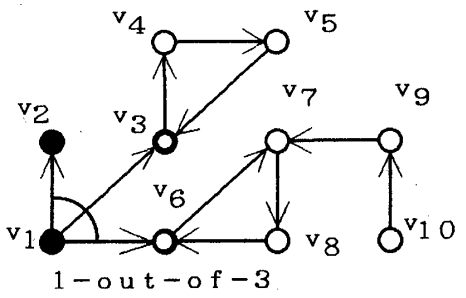

図3．　デッドロックの回復の例

4．あとがき

本稿では，N-out-of-M要求を考慮した分散形システムのモデルを導入し，デッドロックの検出と回復を行う分散形アルゴリズムを提案した．なお，回復アルゴリズムで採用したヒューリスティック解法の性能評価[4]については別の機会に発表の予定である．

文献

[1] Chandy, K. M. and Lamport, L.:"Distributed snapshots: Detecting global states of distributed systems," ACM Trans. Computer Systems, 3, 1, pp.63-75 (1985).

[2] 緒方，杉原，菊野，吉田:"重複データベースシステムにおける分散形デッドロック検出アルゴリズム"，信学技報，AL85-3，pp.17-26 (1985).

[3] 緒方，平川，菊野，吉田:"N-out-of-M要求に基づく分散形システムのおけるデッドロックの回復"，信学技報，AL85-40，pp.17-26 (1985).

[4] Ogata, M.:"On distributed algorithms for deadlock detection and resolution," Master Thesis, Hiroshima University (1986).