2H-2　計算と検査を並行して行う高可用性システムにおけるスケジュール

最所圭三　上林弥彦
(九州大学工学部)

1．まえがき

計算機の信頼性や可用性を上げる一般的な方法として，同じ計算を複数のプロセッサで実行しその結果を比較して処理を進める方法がある。この方法を実際のシステムに応用するには、複数のプロセッサモジュールを適当な通信機構で接続し、各モジュールで同じ計算を冗長に行い互いにその結果を転送し比較しながら処理を進めるようにすることで比較的簡単に実現できる。またデータベースシステムでは処理の効率向上のために複数のトランザクションを並行して処理するため、トランザクションの処理順に矛盾を生じないように並行処理制御を行う。データベースシステムの可用性を上げるために上記の冗長計算を取り入れる場合、入出力，スケジュール（並行処理制御），故障等からの回復処理，… 等の様々な問題を生じる。我々は既に、高可用性システムにおける並行処理制御として計算と結果の比較を並行して処理する方法[3]を，回復処理としてその障害の程度に応じて処理する方法[4]を提案した。

文献[3]では、代表的な並行処理制御方式である２相ロック方式と時刻印方式を拡張して冗長計算と結果の比較を並行して行えることを示したが、その方法を実際のシステムに応用した場合制限が厳しく処理効率の向上が困難であると予想された。本稿では、文献[3]で示した時刻印方式の改良を行ったのでこれについて報告する。

2．問題点

本節では、単一システム上における時刻印方式を文献[3]で拡張した方法を簡単に示しその問題点を指摘する。

トランザクションは図１に示すように読出しと書込みの系列からなる半順序有向グラフで表すことができる。計算結果は書込み操作によってのみデータベースに反映されるため、書込みデータについてのみ検査すればよい。

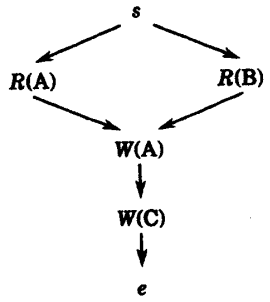


図１ トランザクションの例

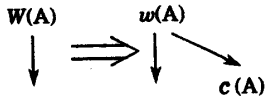


図２ 書込み操作の分解

よって、書込み操作を図２に示すように他のモジュールへの計算結果の転送（w(A)）と結果の比較および実際の書込み（c(A)）に分解する。このとき、｛W(A)→W(B) ならば c(A)→c(B)｝ の条件を加えることで確定したデータを用いて計算したデータについてのみ結果の検査を行うことができる。

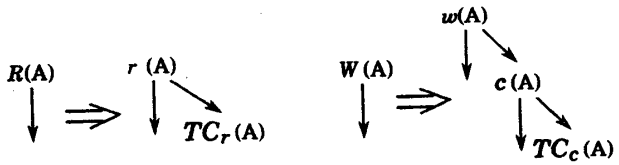
ここで時刻印方式を用いて並行処理制御を行うには、モジュールが異なっても同一のトランザクションに対して同じ時刻印が与えられると仮定すると、以下の条件を満たすことで直列可能性が満足される。

(1) 局所的には通常の時刻印方式の規約を守る。

(2) 大域的には同一の操作を処理するモジュール上で、読出しでは同じトランザクションが書いたデータを読出し、書込みでは同じトランザクション順に書込むようにすればよい。これは、読出し，書込みの両方ともそのときのデータの書込み時刻が一致しているかで検査できる。

上記の条件が守れないときにはそのトランザクションを後退復帰する。(2)で実際の操作と時刻印の検査を並行して行うために読出し，書込み操作を図３に示すように分解する。TCr(A)，TCc(A) が時刻印の検査でトランザクションのコミット前に終了していなければならない。

以上が文献[1]で示した方法である。この方法では、各モジュール上での操作の実行のちょっとした時間的ずれで後退復帰の可能性が増加する。例えば、図４ではモジュールＣのＴ5の書込みにより後退復帰を発生する。もし



(a) 読出し操作の分解　　(b) 書込み操作の分解

図３ 各操作の分解

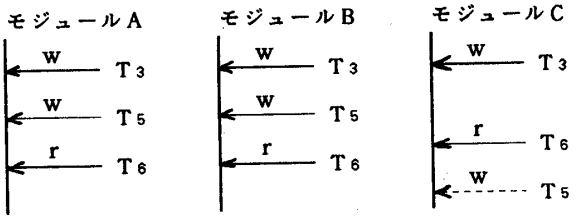


図４ 操作の時間的ずれによる後退復帰

Scheduling of a Highly Available System with Concurrent Execution Computation and Checking

Keizo SAISHO, Yahiko KAMBAYASHI
Kyushu University

モジュールC上のT_6の読出しを遅らせてT_5の書込みを先に実行できれば後退復帰がなくなる。次節ではこの様に操作の実行を遅らせることによって後退復帰の可能性を少なくする方法を示す。

3．操作の実行を遅らせる時刻印方式

本節では、2節で生じた時刻印を用いて冗長計算を行うシステム上での並行処理制御の欠点を操作の実行を遅らせることにより改良する。但し、ここでの方法は大域的な時刻印の不一致による後退復帰の可能性を少なくするだけで、局所的な条件による後退復帰に対しては効力も持たない。基本的には、各操作を行うときにその対象になるデータの時刻印をチェックして、その操作の前に実行すべき操作を予想することで実際の操作を遅らせて後退復帰の可能性を少なくする。例えば、図4のモジュールCのT_6の読出しの時にモジュールAやBの書込み時刻印によりT_5の書込み操作を予想でき、この読出し操作をT_5の書込み操作が終るまで遅らせる。

上記の方法を実現するために、データに通常の読出し時刻印と書込み時刻印の他に、待ち状態を示す待ちリストを導入する。以下に読出し，書込み，時刻印の変更および待ちリストの操作について説明する。ここでは、3重系についてのみ考慮する。まず最初にここで使用する記号を示す。

t : トランザクションの時刻印
tr_A, tr_B, tr_C : 各モジュール上での読出し時刻印
tw_A, tw_B, tw_C : 〃 書込み時刻印
$tr = max(tr_A, tr_B, tr_C)$, $tw = max(tw_A, tw_B, tw_C)$

待ちリストの構造

待ちリストの構造を図5に示す。r/w は操作の種類で r，w の2つの状態を持つ。Req は実際に操作の要求がきているかどうかのフラグで T（要求あり），F（要求なし）の2つの状態を持つ。t_t はその操作が含まれるトランザクションの時刻印を表す。リストに含まれる操作は、1) リストの先頭に位置する，2) Req = T という2つの条件が揃った時点で実行し、通常の方法で時刻印を変更する。

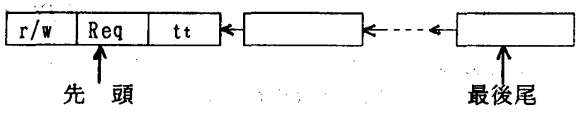


図5 待ちリストの構造

以下に読出しと書込みの手順を示す。

読出し操作

1) t < tw のとき：このトランザクションを後退復帰する。
2) t > tw のとき：
   i) r/w = r, t_t = t のリストが待ちリストに存在するとき：Req ← T にする。
   ii) i)の条件が満たされない時：全モジュールの待ちリストの t_t < t < t_t'（t_t' は t_t の直後のリスト）の条件を満たす場所に
   r/w ← r, Req ← F, t_t ← t
   にしたリストを挿入し、その後自分のモジュール上でのみ Req ← T にする。

書込み操作

1) t < tr のとき：このトランザクションを後退復帰する。
2) t > tr のとき：
   i) t < tw_k（k は自分のモジュール名）のとき：すでに上書きされているのでなにもしない。
   ii) tw_k < t < tw のとき：r/w = w, t_t = t のリストが待ちリストに存在するとき Req ← T する。存在しなければなにもしない。
   iii) tw < t のとき：
      a) r/w = w , t_t = t のリストが待ちリストに存在するとき：Req を有効にする。
      b) a) の条件が満たされない時：全モジュールの待ちリストの t_t < t < t_t' の条件を満たす場所に
      r/w ← w, Req ← F, t_t ← t
      にしたリストを挿入し、その後自分のモジュール上でのみ Req ← T にする。

4．結論

以上、計算と検査を並行して行う高可用性システムでの並行処理制御を時刻印を用いて行う際に、実際の実行を遅らせることによって後退復帰の可能性を少なくする方法を示した。

文献[3]の方法は互いの時刻印を検査することによりモジュール間の矛盾を発見し、矛盾が生じたときに後退復帰するいわば受動的な方法であった。これに対し、本方式では互いの時刻印を検査することによってあるモジュールではまだ実行されていない操作がいずれ実行されるであろうことを予想し、実行を待たせる能動的な方法に変えた。この方法では、モジュール間の微妙なタイミングのずれで起こっていた後退復帰をなくすことができるが、時刻印の検査および待ちリストへの登録までの操作をシステム全体に渡って原子的に行う必要が生じる。よって、後退復帰が希な場合には、この部分の制御のオーバヘッドがネックになり文献[3]の方法より効率が落ちると思われる。

参考文献

[1] J.H.Wensley, M.W.Green, K.N.Levitt and R.E.Shostak : "The Design, Analysis and Verification of the SIFT Fault Tolerant System", Proc. 2nd International Conference on Software Engineering, pp.458-469, 1976.
[2] W.Kim（上林訳）：高い可用性を持つデータベースとデータベース応用システム，bit別冊59年1月，最近のデータベース・システムとその応用，共立出版，pp.176-194, 1984.
[3] 最所，上林：並行処理制御の冗長計算システムへの拡張，信学技報 EC85-53, 1985.
[4] 最所，上林：並行処理制御を行う高可用性システムにおける回復処理，情報処理学会第32回全国大会 2C-3, 1986.