5M-1

リアルタイムデータベースシステムにおけるトランザクションの最適スケジューリング

道本 芳樹　宮尾 淳一　若林 真一　吉田 典可
広島大学工学部

1．まえがき

座席予約システム，キャッシュカードシステムなどのリアルタイムデータベースシステム(RDBS)では，デッドラインをもつトランザクションのスケジューリング[1]を行う必要がある．したがって，RDBSのスケジューラは次の①，②の条件を満足するスケジュールを出力しなければならない．①データベースの一貫性を保証するためスケジュール結果が直列可能(serializable)である．②各トランザクションの処理が予め設定されたデッドラインまでにできるだけ終了する．従来，このようなスケジューラに関する研究は十分に行われていなかった．

本稿では，前述の①，②の条件を満たし，かつトランザクションを複数のプロセッサに割当てるスケジューラに関する基礎的，理論的考察を行う．

2．スケジューリング問題の定式化

まず，図1に示すようなシステムモデルを考える．RDBSで処理されるトランザクションは入力キューに入る．スケジューラはある一定の時間間隔でキューに入っているトランザクションのスケジュールを決定し，各プロセッサに割当てる．以下では，一つの時間間隔におけるスケジューリング問題を(A1)-(A3)の仮定の下で考察する．

(A1)データベースは主記憶に納まるとし，各データのアクセス時間は一定とする．

(A2)プロセッサはすべて同一であり，計算能力，速度はすべて等しい．

(A3)トランザクションを構成する演算ごとにプロセッサの先取り(preemption)を許す．また，一つのトランザクションの演算が複数のプロセッサに割当てられる場合はプロセッサ変更ごとに一定の通信コストC_{com}がかかるとする．

次に，各トランザクションがデッドラインをミスする時間を最小にし，データベースの内容の一貫性を保証するスケジューリング問題MMT(Minimum Missed Time)を定式化する．

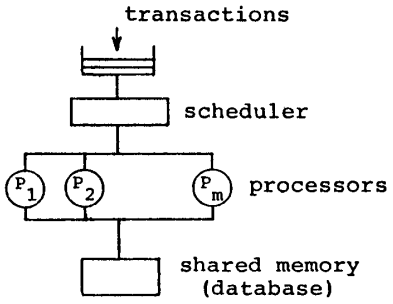


図1　システムモデル

[問題MMT] 以下のような入力が与えられたとき，条件を満足し目的関数MTの値を最小にするスケジュールSを求めよ．

入力：(1)トランザクションの集合$T=\{T_1, T_2, \cdots, T_n\}$，ただし，$T_i=\{T_{i1}, T_{i2}, \cdots, T_{i\ell}\}$ (T_{ik}は演算)．

(2)T_i内でのT_{ij}，T_{ik}間の全順序$\prec_i$．

(3)トランザクションT_iを構成する各演算T_{ik} ($1 \leq k \leq n$)の実行見積り時間E_{ik} ($\sum_{k=1}^{\ell} E_{ik}=E_i$とする)．

(4)T_iのデッドラインd_i．

(5)プロセッサ数m．

出力：各プロセッサP_k ($1 \leq k \leq m$)上でのスケジュールS_kの集合S(S_kはt_{ij}をT_{ij}の処理開始時刻とするとき(T_{ij}, t_{ij})の系列)．

条件：入力で与えられた各トランザクションの全順序$\prec_i$を満たし，出力として得られるスケジュールは直列可能であること．

目的関数：$MT=\sum_{i=1}^{n} \max(e_i - d_i, 0)$

(e_iはT_iの終了時刻を表し，C_{com}は各トランザクションの処理時間の中に含めるとする)．　□

3．プロセッサ数1のスケジューリング

問題MMTにおいて，プロセッサ数m＝1の場合のスケジューリング問題について考察する．このとき，以下の補題1，補題

Optimum scheduling of transactions in a real-time database system
Yoshiki MICHIMOTO, Jun'ichi MIYAO, Shin'ichi WAKABAYASHI, and Noriyoshi YOSHIDA
Hiroshima University

2が成立する[2].

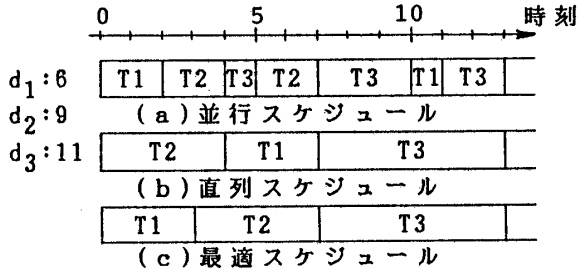
[補題1] プロセッサ数が1の場合の任意のスケジュールSは,目的関数MTの値を増加させることなく,先取りを許さない直列なスケジュールS'に変換することができる. □

補題1より,プロセッサ数が1の場合,問題MMTの最適解は先取りを許さない直列なスケジュールの中に存在することになる(図2(a),(b)).この場合,直列可能性について考察する必要はない.

[補題2] プロセッサ数が1で,かつ先取りを許さない任意のスケジュールSにおいて,$e_j<e_i, E_j>E_i, d_j>d_i$を満足する$T_i, T_j$が存在するならば,目的関数MTの値を増加させることなくT_iとT_jの実行順序を入れ換えたスケジュールS'に変換できる(図2(b),(c)). □

補題1,補題2より,プロセッサ数が1で,かつ実行時間の小さいトランザクションほど小さいデッドラインをもつならば,次のアルゴリズムで最適解を求めることができる.

[アルゴリズムA] 各トランザクションをデッドラインの非減少順にソートし,この順序でプロセッサに割当てる(図2(c)). □



(a)並行スケジュール

(b)直列スケジュール

(c)最適スケジュール

図2 プロセッサ数1のスケジューリング

[定理1] 問題MMTにおいて,m=1かつトランザクションの実行時間が小さいものほどデッドラインが小さいならば,アルゴリズムAは$O(n\log n)$で最適解を求める[2].ここで,n=|T|である. □

4. プロセッサ数2のスケジューリング

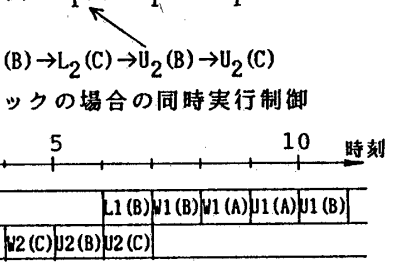
トランザクションの同時実行制御を考える場合,実行の順序関係は図3(a)に示すような有向グラフで表すことができる.ただし,図3(a)では実行順序決定に係わるロック,アンロック演算のみを抜き出している.同図において,1つのトランザクション内の有向枝は入力で与えられる全順序を表しており,異なるトランザクション間の有向枝は,それらのトランザクション間でデータの競合が起こった場合にどのような順序で実行するかを表している.例えば,$U_2(B)$から$L_1(B)$への有向枝は,$U_2(B)$の実行が終了してから$L_1(B)$の実行が開始できることを表している.

以下では,m=2かつT={T_1, T_2}の場合について考察する.

まず,各トランザクションが2相ロックに従う場合,次のアルゴリズムで最適解を求めることができる.

[アルゴリズムB] 競合が起こっている各データXに対して$U_1(X)$から$L_2(X)$への有向枝を設ける.T_1, T_2それぞれに対し最後のアンロックまでの最長パスを求め,目的関数の値を計算する.$T_2 \rightarrow T_1$の方向に同様の操作をし,目的関数の値が小さい方の有向枝を採用する.次に,前述の操作で求まった先行関係を保存して,T_1, T_2の各演算をそれぞれプロセッサP_1, P_2に割当てる. □

[例1] 2つのトランザクションT_1, T_2の実行時間とデッドラインがそれぞれ,E_1=6,E_2=7,d_1=9,d_2=10と与えられたとき,アルゴリズムBを適用すると図3(b)のような最適スケジュールが得られる.このときの目的関数の値はMT=2+0=2である. □



(a)2相ロックの場合の同時実行制御

(b)最適スケジュール

図3 2相ロックの場合のスケジューリング

[定理2] 問題MMTにおいて,各トランザクション内の要求が2相ロックに従う場合,アルゴリズムBは$O(|T_1|+|T_2|)$で最適解を求める[2]. □

[定理3] 問題MMTにおいて,入力されるT_1, T_2内の各演算と全順序が等しい(デッドラインは異なってよい)ならば$O(|T_1|)$で最適解が求まる[2]. □

5. 今後の課題

本稿では,問題MMTをm=1,及びm=2かつn=2の場合について考察した.現在,m≦nかつn≧3の場合について考察中である.

文献 [1]P.A.Bernstein, et al. :"Concurrency Control and Recovery in Database Systems", Addison-Wesley (1987). [2]道本芳樹:"時間的制約をもつトランザクションの最適スケジューリング問題", Hiroshima Univ. ECS Lab.,Tech. Rep. 89-07 (1989).