5E-10

先行制御方式におけるオーバヘッドの解析

小林　真也[1]，渡辺　尚[2]，中西　暉[1]，手塚　慶一[1]
[1]大阪大学　[2]徳島大学

1．はじめに

分散処理は，集中処理に比べ経済性，拡張性，変更容易性，高速性，信頼性など数多くの利点を持つ．上述のような多くの利点の中で，特に大規模なジョブの高速処理を狙った分散処理の部分概念として並行処理がある．

並行処理は，処理能力を持つ構成要素（以下，プロセッサと呼ぶ）をほとんどすべて並列に動作せることにより処理効率の向上を達成しようというものである．このため並行処理の対象となるジョブは当然高い並列性を持つジョブであるが，これには事前に処理計画がたてられるスケジュール可能なジョブと確率的不確定要素を含むために処理計画をたてることができないスケジュール不可能なジョブの２つがある．前者の例としては画像処理，行列計算等があり，これらのジョブの実行を行なう並行処理システムはすでに多数発表され実用システムも存在する．一方，待ち行列網シミュレーションや分散データベースのようにスケジュール不可能なジョブでは，プロセッサ間での通信がいつ発生するかわからないため，プロセッサ間における処理順序に矛盾なく処理を進めることは容易ではなく，プロセッサ間の同期法が問題となる．この同期法の１つに一時的な矛盾を許し，キャンセル処理によりそれを解消する先行制御方式がある．しかしながら，処理速度低下の原因となる分散化オーバヘッドならびにプロセッサへのジョブ分割割当法に関しては十分な知見が得られていないのが現状である．

そこで，本稿ではスケジュール不可能なジョブを実行する並行処理システムの同期法として処理能力がプロセッサ台数に対して線形に増加する先行制御方式[4]をとりあげ，処理速度低下に深く関係する矛盾発生について検討を行う．

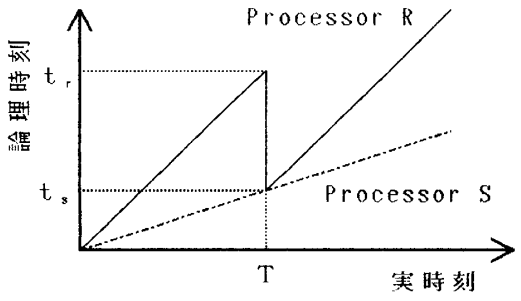

図１　先行制御方式

2　先行制御方式

2．1　同期法

先行制御方式では，以下に示すアルゴリズムにしたがって処理を行なう．

① 各プロセッサは他のプロセッサとまったく独立に処理を行なう．

② 送信側プロセッサはメッセージに自分自身の論理時刻を付加し送信する．

③ 受信側プロセッサはメッセージを受け取ると，メッセージの持つ論理時刻t_mと現在の論理時刻tを比較し，

◎ $t_m \geqq t$の時

論理時刻がt_mに達するまでこのメッセージの処理を行なわない．

◎ $t_m < t$の時

受信側プロセッサは処理を進めすぎたこととなり矛盾が発生する，そこで受信側プロセッサは進みすぎた処理を取り消し論理時刻をt_mまで戻し（キャンセル処理）再び処理を再開する．

図１は先行制御を行なった際のプロセッサの実時刻と論理時刻の関係を示したものである．図１において送信側プロセッサＳは実時刻Ｔに論理時刻t_sをもつメッセージを送る．これを受信するこのときの受信側プロセッサＲの論理時刻t_rは$t_r > t_s$であるから矛盾となる．そこでプロセッサＲはその状態を論理時刻t_sのときの状態に戻し矛盾を解消し，再処理を行なう．

先行制御方式では受信側プロセッサはメッセージを受け取るまで送られてくる時刻を知ることはなく，メッセージが送られて初めて矛盾であるかどうかの判断を行う．このように，先行制御方式は処理過程に一時的な矛盾は発生するが，矛盾が発生すればキャンセル処理を行ない矛盾を解消することで，最終的な処理過程から矛盾を排除している．

2．2　矛盾状態発生率

先行制御方式はプロセッサが処理を中断するアイドル状態がないため並列性は高いと考えられる．しかしながら，矛盾状態はそれまで余分な処理を行なっていたことを意味し，矛盾解消のためにキャンセル処理を行なわなければならず，処理速度は低下する．そこで，矛盾発生率を求め先行制御方式の処理速度改善について検討する．

矛盾発生率を以下のように定義する．

$$(\text{矛盾状態発生率}) = \frac{(\text{矛盾状態発生回数})}{(\text{メッセージ送信回数})}$$

解析対象となるモデルを以下のように仮定し，こ

An Analysis of Overheads on Advance and Parallel Processing Control Scheme
Shinya KOBAYASHI[1], Takashi WATANABE[2], Hikaru NAKANISHI[1], Yoshikazu TEZUKA[1]
[1]OSAKA UNIVERSITY　[2]TOKUSHIMA UNIVERSITY

のモデルの矛盾状態発生率を求める．

① システムは送信側プロセッサSと受信側プロセッサRの2台のプロセッサからなる．
② 送信側プロセッサが1イベント終了毎にメッセージを送る確率は$P_s=0.5$である．
③ 各プロセッサの1イベントに要する実時間は，送信側プロセッサSにおいては平均$\lambda_s=1/p_s$，受信側プロセッサRにおいては$\lambda_r=1/p_r$の指数分布にしたがう．
④ 各プロセッサが1イベントで更新する論理時間は，送信側プロセッサSにおいては平均$\mu_s=1/l_s$，受信側プロセッサRにおいては$\mu_r=1/l_r$の指数分布にしたがう．
⑤ メッセージの伝搬遅延はない．
⑥ メッセージの送受信があると，送信側プロセッサSと受信側プロセッサRの論理時刻は一致するとする．

このようなモデルに対し以下のような手順で矛盾状態発生率を求める．

送信側プロセッサSは平均$1/P_s$個のイベント毎にメッセージを送る．つまり，実時間で$T_s=1/(\lambda_s\times P_s)$毎にメッセージを送る．実時間$T_s$の間に送信側プロセッサSにK個のイベントが発生する確率$P_{sk}(T_s)$は，

$$P_{sk}(T_s)=\frac{(5p_s)^k}{K!}\exp(-5p_s).$$

$$\sum_{k=0}^{\infty}P_{sk}(T_s)=1.$$

となる．また，プロセッサRでL個のイベントが発生する確率P_{rl}も同様にして求めることができる．

したがって，実時間T_s秒の間にプロセッサSでK個，プロセッサRでL個のイベントが起こる確率$P(K,L\mid T_s)$は，

$$P(K,L\mid T_s)=P_{sk}(T_s)\times P_{rk}(T_s).$$

となる．

また，プロセッサSにイベントがK個ある時の論理時間の分布の密度関数$p_{sk}(t)$は，Kステージアーラン分布となり，

$$p_{sk}(t)=\frac{Kl_s(Kl_st)^{k-1}}{(K-1)!}\exp(-Kl_st).$$

となる．同様にプロセッサRにイベント数がL個ある時の論理時間の分布の密度関数$p_{rl}(t)$もLステージアーラン分布となる．

図1からわかるように$t_r>t_s$の時に矛盾となり，プロセッサS，プロセッサRのイベント数がそれぞれK，Lである時に矛盾状態となる確率$P_{kl}(t_r>t_s)$は

$$P_{kl}(t_r>t_s)=\int_0^{\infty}\{p_{sl}(\tau)\int_{\tau}^{\infty}p_{rk}(x)\,dx\}\,d\tau.$$

となる．したがって，矛盾状態発生率P_cは，

$$P_c=\sum_{l=1}^{\infty}\sum_{k=1}^{\infty}\{P(K,L\mid T_s)\times P_{kl}(t_r>t_s)\}.$$

となる．

3．結果

解析の結果，矛盾状態発生率P_cはλ_rとλ_sの比（λ_r/λ_s）（以下，これを”実時間比”と呼ぶ），またμ_sとμ_rの比（μ_s/μ_r）（以下，これを”論理時間比”と呼ぶ）によって決定されることがわかった．図2はx軸に実時間比（λ_r/λ_s），y軸に論理時間比（μ_s/μ_r），z軸に矛盾発生率P_cをとり3次元空間上でこれら2つのパラメータと矛盾発生率の関係を図示したものである．

4．結び

本稿ではスケジュール不可能なジョブの実行を行なう並行処理システムの同期法の1つである先行制御方式に対し，処理速度低下の原因となる矛盾の発生につての解析を行なった．これと並行し他の同期法である制御メッセージ方式におけるオーバヘッドの検討も行い，各方式における処理分配についての検討も行なっているが，これらについては紙面の都合もあり他の機会にゆずる．今後，これらの結果を用いて，実際のシステムへの応用を検討していく予定である．

参考文献

[1] 古川,小林,中西,手塚:信学会システムのモデリングと性能評価時限研究専門委員会第1回研究会資料,pp.13-18(1988)
[2] 小林,渡辺,真田,手塚:確率的要素を取り扱うためのメッセージパッシングの拡張,信学技報,COMP87-59,pp.41-49(1987)
[3] 小林,渡辺,真田,手塚:メッセージパッシングによる並行処理系のプロセス間関係の取扱について,信学会昭和63年春季全国大会,D-380,(1988)
[4] 佐藤,中西,真田,手塚:並行処理形シミュレータD-SSQ,信学論, J69-D,No.3(1986).
[5] 渡辺,中西,真田,手塚:規制先行制御方式を用いた非同期ジョブ並行処理システムの処理能力解析,信学論(D),J69-D,No.10(1986).

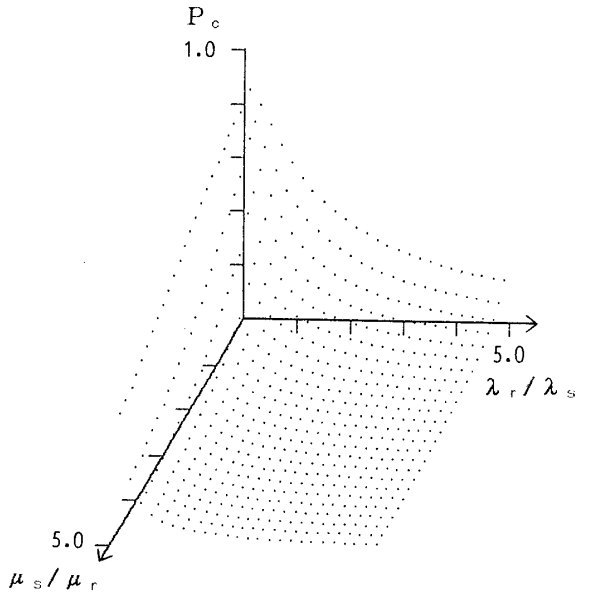

図2 実時間比 vs. 論理時間比 vs. 矛盾発生率