7G－1

デッドロック検出手法紹介

有村　雄二[1*]　宮地　浩司郎[1*]　木下　光幸[1*]　浅間　幸夫[2*]

1*　富士通　㈱　　2*　㈱　両備システムズ

1. はじめに

マルチプログラミングシステムでは、ファイル等の共用資源をアクセスする場合には、他のプログラムからのアクセスを一時的に防ぐために排他制御を行う必要がある。排他制御を行う場合、順序性や規則を守らないと資源を相互に取り合う事象が発生し、デッドロック状態に陥りシステムが停止することがある。

システムを構成するプログラムが増大し複雑化するにつれ、このデッドロックに陥る可能性は急速に高まっている。

デッドロックに陥る原因の種類については後述するが、多くの場合その影響はシステム全体に及び業務に多大なる支障をきたす。

また、近年増大しつつあるＬＣＭＰ(Loosely Coupled Multi Processor) 構成の場合、全システムに影響を及ぼすこともある。

今回我々は、富士通最上位汎用オペレーティング・システムＯＳⅣ／Ｆ４　ＭＳＰにおいてデッドロックを未然に検出できる手法を考案・実現したので、ここにその検出手法を紹介する。

2. デッドロックの原因と特性

デッドロックが起こる原因として、以下のケースがあることが分かっている。

① ＬＣＭＰ構成において２台以上の共用装置に対し占有を要求する。（相互に相手システムで占有している資源を要求すると両システムは待ち状態になる）
② ＬＣＭＰ構成において共用装置を占有した状態でその装置以外の装置に対し入出力を実行する。（①同様に両システムが入出力実行待ち状態になる）
③ 他タスクを強制停止した状態で、資源を要求する。（強制停止させられたタスクが要求した資源の占有者であった場合、永久に資源は獲得できない）
④ 複数の資源を異なる順序で要求する。（タスク間でお互いの資源獲得待ち状態になる）

①②はシステム間、③④はシステム内におけるデッドロック発生のケースである。

いずれのケースにおいても、システム間あるいはプログラム間で占有処理が同時に発生しないとデッドロックは発生しない。しかし、資源の占有時間は全体の処理に比べて、極めて短いため、まれなタイミングでしか発生しない。これは、短期間のテストでは検出できない原因の一つとなっている。

また、システムを構成するプログラムが増大し、部品化が進行すると、一つのプログラム内で排他制御が規則どおりに行われていたとしても、これらが組み合わされシステム全体として動作した場合、結果的に規則に違反してしまうケースが増えてきている。

3. デッドロック検出の実現方法

これまで述べたとおり、複数の資源の占有を要求すれば、デッドロックを起こす危険性をはらんでいると言える。

従って、二つ以上の資源の獲得が要求された場合に、要求元の状態や環境を調べれば、その要求によりデッドロックに陥る可能性があるかどうかを検出することができる。

資源の獲得要求とは以下の事象があり、それぞれの事象が発生した時、デッドロック検出プログラムに制御が渡るような機構を設けた。

－ 共用装置に対する占有要求
－ システム内の資源の獲得要求
－ 入出力の実行

それぞれの事象が発生した時、デッドロック検出プログラムではデッドロックとなる原因に対応して、以下のチェックを行っている。

－ 共用装置に占有要求を行っているタスクが、他ボリュームを既に占有していないかチェックする　……ケース①
－ 入出力要求を行っているタスクが、他ボリュームを既に占有していないかチェックする　……ケース②
－ 資源を要求したタスクが他のタスクを停止させていないかチェックする　……ケース③
－ 二個以上の資源要求を出した時、過去に逆の順序で要求した事がないかチェックする　……ケース④

以下にデッドロック検出機構のしくみを示す。

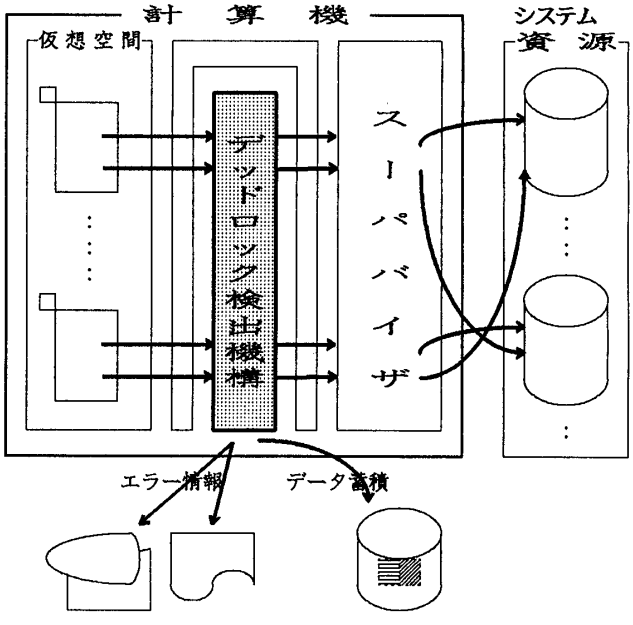

図　　デッドロック検出機構のしくみ

4. 効果

－ 本プログラムは、デッドロックが発生した事を通知するのではなく、デッドロックに陥る可能性のあることを警告するものである。従って、以下のメリットがある。

(1) 同時に占有処理が起こらなくても検出可能なため、テストに時間をかける必要はない
(2) ＬＣＭＰでの運用を行う必要がなく、単一システムでも共用資源に対するデッドロックを検出できる

－ 本プログラムは特別な操作は必要とせず、システム運転中に動作させておくだけでよい。したがって、利用者は本プログラムの存在を意識することなくテストを実施することができる。

Introduction of a method to detect Deadlock

Yuji ARIMURA[1*], Kojiro MIYACHI[1*],
Mitsuyuki KINOSHITA[1*] , Sachio ASAMA[2*]

1*　FUJITSU　Ltd.　　2*　RYOBI SYSTEMS Co.,Ltd.