# Datarolマシンの資源管理方式に関する検討
## − プロセス状態検出方法と状態変化制御機構 −

2L−9

星出 高秀　日下部 茂　谷口 倫一郎　雨宮 真人
九州大学総合理工学研究科

## 1　はじめに

動的データ駆動方式を基礎とした並列計算機Datarolマシンは，関数型プログラムの実行時に生じる小粒度の多数のプロセスを並列に実行し，超多重処理環境を実現する[Ama88]．Datarolプロセッサは，従来のデータ駆動型計算機の問題点を解決する1つの手段として，同一インスタンス内のデータを共有するために，関数呼び出し時に生成されるプロセス(以下，インスタンスと呼ぶ)毎に1枚のレジスタファイルを割り付け，冗長なデータフローを削除する．

しかし，プログラムが高い並列性を持つ場合，利用可能なレジスタファイル数を考慮した並列展開制御と効率的レジスタファイル管理を行なう必要がある．Datarolプロセッサでは，関数レベルの並列展開を制御し，レジスタファイルのスワップ管理を行なう[Hoshi91]．

これまでのシミュレーションによるとレジスタファイルのスワップ管理には改善の余地があり，その効率化にはいくつかの方法[Hoshi91][Kusa][Hoshi92]がある．本稿では，スワップ処理回数の削減による効率化について検討し，シミュレーションによる評価を行なう．

## 2　レジスタファイル管理の効率化

現在コンパイラは，関数内での部分評価により，可能な限り処理を進める(以下，eagerな関数活性化と呼ぶ)ためのコードを生成する．しかし，有限のレジスタファイル下でのプログラム実行では，「eagerな関数活性化により，利用可能なレジスタファイルが不足し，スワップ処理が頻発する」という問題が起こる．

この問題の解決策として，条件付関数活性化が考えられる．一般に，関数が活性化するタイミングは，1)引数受取り時と2)結果値受取り時である．1)に関しては，全ての必須引数(関数の実行に必ず必要となる引数)が到着してから活性化する，2)に関しては，他のcalleeからの結果値が揃ってから活性化する，という条件付活性化でスワップ処理回数を削減する．

### 2.1　引数受取り時における関数活性化制御

図1に関数Fの簡略化したDatarolグラフを示す．本節では，関数Fを例としてcalleeが引数受取り時に抱える問題点を示し，コンパイラによる解決方法を述べる．

関数Fを呼び出すインスタンスは，引数データx,yの存在に関係なく関数Fを活性化する．そのため実行時に，関数Fは活性化しレジスタファルを割り付けられたものの，引数x,yが到着するまでコードブロックB以下の実行が不可能な状態になる．このように，eagerな関数呼び出しにより，無駄なレジスタファイル割り付けが生じ，スワップ処理回数が増加する可能性がある．

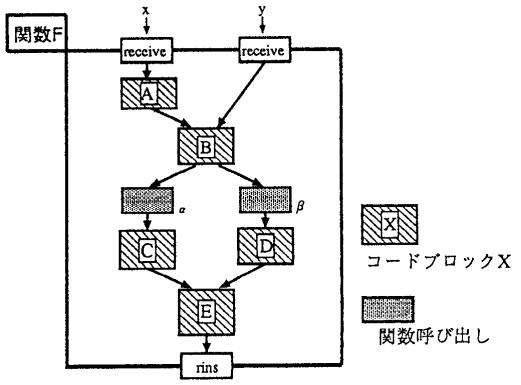



図1: Datarolグラフの抽象例

上記の問題点を解決するために，関数活性化条件を「全ての必須引数の到着」とする．この条件により，活性化されたインスタンスには実行可能なコードブロックが多数存在するので，無駄なレジスタファイル割り付けが削減でき，スワップ処理回数を抑えることができると思われる．(関数Fに関しては，必須引数x,yの確定後に関数Fが活性化される．)

この条件付活性化の実現方法として，callerにおける必須引数確定後の関数呼び出し実行が考えられる．calleeにおける必須引数の解析と，callerの必須引数の確定まで関数呼び出しの実行を行なわないようなコードの生成をコンパイラで行なう．

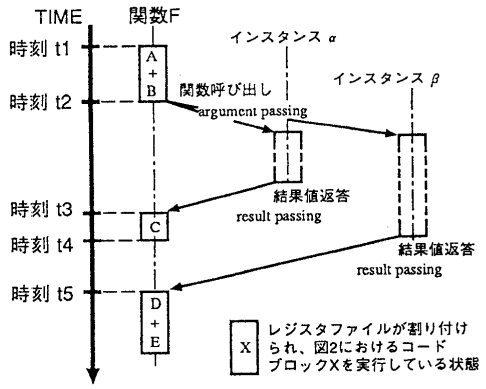
### 2.2　結果値受取り時における関数活性化制御

本節では，関数Fを例としてcallerが結果値受取り時に抱える問題点を示し，その1つの解決策としコンパイラ支援のもとハードウェアによる解決策を示す．

図2に関数Fの状態変化図を示す．図2のように，関数Fは4回の状態変化を繰り返す．このため，時刻t1,t3,t5の3ヶ所でスワップ処理が生じる可能性がある．関数Fのように，複数の関数適用をもつインスタンスは，頻繁に状態変化を繰り返すので，同一インスタンスに対するレジスタファイル割り付けが頻繁に生じ，スワップ処理回数の増加につながる可能性がある．

上記の問題点を解決するために，関数活性化の条件を「他calleeからの結果値返答による待ちなし」とする．この条件

A Consideration on Resource Management of Datarol Machine
− Static detection of process state and State transition control by compiler and hardware −
Takahide HOSHIDE, Shigeru KUSAKABE, Rin-ichiro TANIGUCHI, Makoto AMAMIYA
*Department of Information Systems, Graduate School of Engineering Sciences, Kyushu University*




時刻 t1：活性化．時刻 t2：コードブロック A,B の実行後，関数呼び出しによりインスタンス $\alpha$ と $\beta$ を活性化．時刻 t3：インスタンス $\alpha$ からの結果値受取りにより活性化．時刻 t4：コードブロック C の実行を終え，サスペンド．時刻 t4：インスタンス $\beta$ からの結果値受取りにより活性化．

図 2: 関数 F の状態変化図

により，結果値受取り毎の caller の状態変化を防ぐことができ，インスタンスの状態変化回数は減少しスワップ処理回数を抑えることができると考えられる．(図 2の関数 F の状態変化図においては，時刻 t3 での活性化がなくなり，スワップ処理は時刻 t1 と t5 の 2ヶ所に抑えられる．)

この条件付活性化は，以下のようにコンパイラとハードウェアによって実現できる．コンパイラにより，caller が複数の関数呼び出しを持つならば caller はそれら複数の callee からの結果値受取りの同期をとる (図 3を参照)，callee が多値を返すならば callee は結果値を一括して返す，というコードを生成する．そして，実行時における caller 側での複数の callee からの結果値受取りの同期処理をハードウェア (以下，状態変化制御機構と呼ぶ) を用いて行なう．

状態変化制御機構をインスタンス制御ユニット [Hoshi91] 内に設け，複数の callee からの結果値の同期をとり，結果値データは直接外部メモリへ書き込む．データの整合性を保つため，外部メモリへの書き込みは，通常のスワップ処理と同一のストリームで処理する．

### 2.3 プロセス状態検出方法

現在インスタンスの状態検出は，コンパイラ支援のもとでハードウェア・カウンターを用いて動的に行なっている [Kusa]．しかし，条件付関数活性化による効率化を行なうと，任意の関数がサスペンドするタイミングの静的な決定が可能になる．(関数 F は，図 3のようにコンパイルされる．)

この方式により関数内の部分実行を抑えることにより，サスペンド検出が静的に行うことができ，実行時における状態検出のオーバーヘッドとハードウェア・カウンターの除去が可能である．

## 3 シミュレーション評価

本稿で述べた効率化の有効性をシミュレーションにより検討する．シミュレーションには，分割統治法によりフィボナッチ数列を求めるプログラム fibo(20) を用いた．

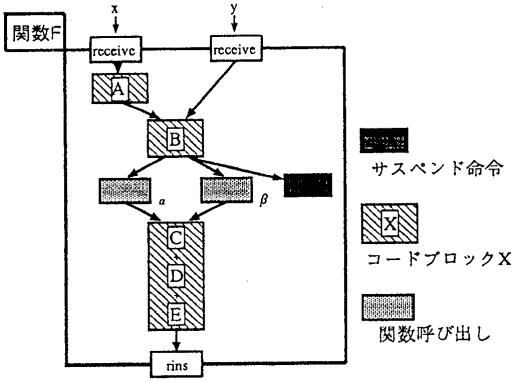



図 3: サスペンド命令を埋め込んだ Datarol グラフ

表 1 において，fibo(20) を 4 プロセッサで実行した時，本方式と従来方式との実行時間を基とした比較を行なう．レジスタファイル数をパラメータとし，レジスタファイルを必要なだけ用意したときの実行時間を 1 とした場合の実行時間比を示す．この表より，本方式は，レジスタファイルのスワップ管理のオーバーヘッドの削減に有効であると考えられる．また，レジスタファイル数が 64,128 のとき，本方式の実行時間比が 1 以下を示している．これは，“スワップなし” の場合，コンパイラによる必須引数の解析と状態変化制御をしていないため，見かけ上アクティブなインスタンス数により負荷が高くなり，パイプラインの乱れが生じたためである．

表 1: 本方式と従来方式との比較

| レジスタファイル数 | 8 | 16 | 32 | 64 | 128 |
|---|---|---|---|---|---|
| 本方式 | 2.69 | 1.62 | 1.03 | 0.94 | 0.92* |
| 従来方式 | 4.15 | 2.08 | 1.19 | 1.00 | 1.00* |

*印は，スワップ処理が行なわれていないことを示す．

## 4 まとめ

Datarol プロセッサにおけるレジスタファイル管理方式の効率化について述べ，シミュレーションによる評価を行なった．シミュレーションにより，条件付関数活性化方式の有効性を確認した．今後は，節度ある並列展開についての検討を行なっていく．

## 参考文献

[Ama88] 雨宮真人，“超並列多重処理のためのプロセッサアーキテクチャ”，情報処理学会「コンピュータアーキテクチャ」シンポジウム，pp.99- 108, 1988.

[Hoshi91] 星出高秀，他 “並列計算機 Datarol マシンにおける資源管理と負荷制御方式”，信学技法，CPSY91-7, pp.25–32, 1991.

[Kusa] 日下部茂，他 “Datarol マシンの資源管理方式”，情処研報，ARC91-5, pp.37–44, 1991.

[Hoshi92] 星出高秀，他 “Datarol マシンの資源管理方式に関する考察”，情処第 44 回全大,7D-2,pp.117–118, 1992.