計算機アーキテクチャ 130－10
（1998. 8. 5）

投機的実行のためのデータ予測可能性

岩田 靖　安里 彰　新井 正樹　小沢 年弘　木村 康則

プログラム中に存在するデータ依存の関係にある命令に対して，データ（値）を予測して投機的に実行する値予測に関する研究が近年始まっている．本稿では選択的値予測と呼ぶ値予測の新しい概念を提案する．これは，値依存の関係にある命令を分類しタスクレベルの投機的実行に効果のある命令のみに対して値予測を行うものである．さらに，効果的に値予測を行うために値予測による利得という概念を導入し，ベンチマークプログラムに対する値予測の効果をスケジューリング，利得，予測正解確率の観点から議論する．簡単なシミュレーションをSPECint92 のベンチマークプログラムに対して4PE で投機的実行を行った結果，約10.2% の命令に対して値予測を行えば良いことがわかった．さらに，差分に基づく簡単な予測機で値予測を行うことにより，59.5% の予測正解確率を達成し，実行時間で平均16.2% の短縮が得られた．

Predictability of Data Values for Speculative Execution

YASUSHI IWATA, AKIRA ASATO, MASAKI ARAI, TOSHIHIRO OZAWA,AND YASUNORI KIMURA

Recently, the study of the speculative execution based on data value prediction is begging at a fundamental level. This paper proposes a new idea of the value prediction which is called Selective Data Value Prediction(SDVP). By the idea of SDVP, we categorize the data dependency type and select the candidate of the prediction data. The candidate of the prediction data is only having a good effectiveness in the sense of task level speculative execution. We also introduce the idea of "gain" for value prediction and discusses the effectiveness of our proposed SDVP in the sense of scheduling of speculative execution, "gain" and value prediction probability. Our preliminary simulation demonstrates that only 10.2% values should be predicted for the speculative execution based on the SDVP criteria for SPECint92 benchmark programs. We achieve 16.2 % reduction of execution time for these benchmark programs by simple predictor. The prediction accuracy of the predictor is about 59.5% in average.

1. はじめに

命令レベル並列度 (ILP) を向上させるために，分岐予測方式が提案されている．分岐予測は分岐命令の分岐方向を予測し，分岐方向が確定する前に後続命令をパイプラインに投入する．このように分岐予測は，制御依存がある複数の命令に対して，分岐方向を予測することで投機的実行を行う命令レベルの投機的実行と理解することができる．

一方，プログラムには制御依存の他にデータ依存が存在する．これは，ある命令で定義されるデータを後続の命令が使用することにより生じる依存関係である．このデータ依存が存在する限りは後続命令を実行することはできない．

近年，データ依存の存在する命令において，そのデータを予測して投機的に実行する値予測の研究が始まっている[1]．しかし，データ依存のある全ての命令に対して値予測を行うのは資源の点から考えても効率的な方法ではないと考える．

本稿では，選択的値予測という概念を導入する．これは，値予測を行うことにより効果のある命令とそうでないものを区別し選択的に値予測を行い，タスクレベルの投機的実行を行うものである．

以降の章では，まず，データ依存が存在する命令のうち，値予測の対象とする命令の定義を行う．そして，選択的に値予測を行う命令においても，より実行速度の向上に効果のある命令とそうでない命令を区別するために「利得」という概念を導入する．さらに，実行モデルを定義して実際のベンチマークプログラムによるシミュレーション結果を示し，選択的に値予測を行った場合にどれだけ性能が向上するかということを，利得と予測確率の側面から議論する．

2. 選択的値予測による投機的実行の基本概念

2.1 制御依存とデータ依存

プログラムには制御依存とデータ依存が存在する．制御依存とは，分岐命令などのプログラムの実行を制御する命令などにより発生する．また，データ依存とは当該命令で使用するデータが，その命令に先行する命令で定義されるような場合に発生する．

新情報富士通研
RWCP Multi-Processor Computing Fujitsu Laboratory

図1に制御依存とデータ依存を図示する．制御依存が

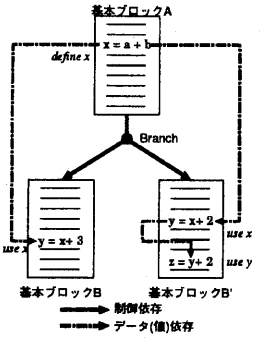

図1　制御依存とデータ依存

存在すると，その制御依存が解消されるまで，それ以降の命令を実行することができない．この問題を解決する技術として様々な分岐予測方式が提案されている．そして分岐予測により分岐先タスクを知り，投機的に後続タスクの演算を実行する方法が提案されている[2)]．

それに対してデータ依存に関しては近年，データ予測を行うことにより，その依存関係を解消し，投機的に後続の命令を実行しようとする研究が始まっている[1)]．

2.2　選択的値予測

実行する全ての命令に対して，その命令が必要とするソースオペランドの値を予測して実行するのは資源の点から考えても極めて非効率的な方法である．

我々は選択的値予測という概念を提案する．選択的というのは，全ての命令のソースオペランド値を予測するのではなく，値予測を行うことによりタスクレベルの投機的実行に効果のある命令のみを予測するものである．なお，本稿では，“タスク”を単一の基本ブロック，もしくは高々数個の基本ブロックからなる基本処理単位と定義する．次に，選択的値予測の選択基準である，「タスクレベルの投機的実行に効果のある命令」を具体的な例により説明する．

タスク間に存在する命令の依存関係を図2に示す．図2において，タスク A とタスク B は図1に示すような制御依存を越えたふたつのタスクであり，分岐予測によりタスク B が実行されると判断されたものとする．そのためタスク A は処理要素 A で処理を行うのと同時にタスク B も投機的に処理要素 B で処理を開始する．本稿では，投機的実行を行う“処理要素”という用語は，プロセッサなど特定のものに限定せずに，プロセッサ内の演算モジュールなども含めた広い意味のプロセッシングエレメントという意味で用いることとする．図2の例では，タスク B 内の命令 (1) で利用されるデータ y は，タスク A の命令 [b] で定義される．同様に，命令 (3) で利用されるデータ x はタスク A の命令 [a] で定義される．さらに，タスク B 内の命令 (2) の z は同じタスク

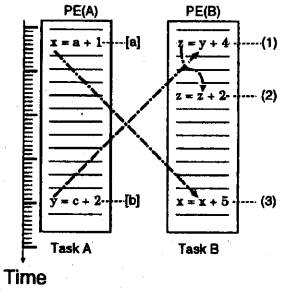

図2　命令間の依存関係

内の命令 (1) で定義されている．

タスク B を投機的に実行する場合，投機的に実行するタスク内に存在する命令が必要とするソースオペランドは，次の3種類に分類される．

1. 自タスク内で定義されるもの．（命令 (2) の z）
2. 他タスク内で定義されるもののうち，
   - 2.1 投機的にその命令を実行する時までにデータが確定しているもの．（命令 (3) の x）
   - 2.2 投機的にその命令を実行する時でも，まだデータが確定していないもの．（命令 (1) の y）

上記のような依存関係のある値のうち，2.1 のカテゴリの命令は，当該命令を実行する時点でデータが確定しているため，予測する必要はない．また，自タスク内で定義される値を予測することは，タスクレベルの投機的実行の性能向上には効果がない．

しかし，もし 2.2 のカテゴリに含まれる命令において，その命令で必要なデータが他タスク内で最終的に定義されるのに先行してそのデータを予測することが何らかの方法でできれば，データ依存を越えてタスクレベルでその命令を投機的に実行することが可能になる．以上のことから選択的値予測の選択基準である「タスクレベルの投機的実行に効果のある命令」を上記の 2.2 のカテゴリに含まれる命令の値とする．すなわち，選択的値予測の対象となる命令は，次のように定義される．

- 当該命令の演算で必要なデータが自分以外のタスクで定義されていて，その演算で必要なデータを定義する先行命令がまだ演算を終了していない場合．

2.3　選択的値予測による利得

プログラムの実行時間の短縮に値予測がどれほど寄与しているかを調査するため，選択的値予測により得られる利得という尺度を導入する．図3に利得の概念を示す．

図3において，タスク B の z を定義する命令は 値 y を必要とし，それは，タスク A において定義される．もし，図3(a) に示すように，値予測を行わずにタスク B を投機的に実行する場合，タスク B の z を定義する命令は，タスク A の y を定義するが命令が発行されるタイミング以降でなければ発行することはできない．それ

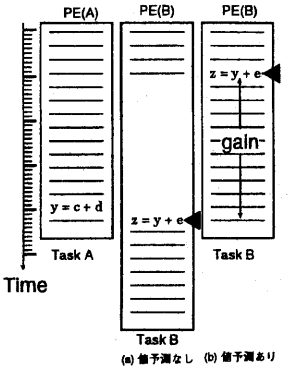

図3 値予測を行うことによる利得

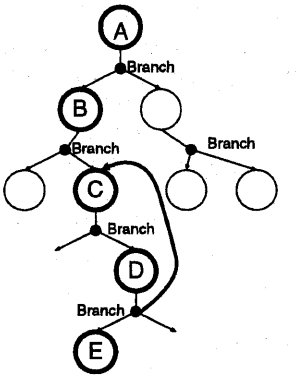

図4 プログラムフローモデル

に対して，図3(b)に示すように，値予測を行う場合，当該命令はデータ依存に縛られずに命令発行が可能になる．すなわち，値予測による利得とは，値予測をせず，依存する値が確定するまで待った場合の命令発行時間時刻と，値予測を行って投機的実行を行った場合の命令発行時刻の差と定義する．

値予測により投機的に実行された全ての命令には利得が定義され，その各々に対して予測正解確率が定義される．このことから，値予測に基づく投機的実行を効率よく行うためには，

$$\sum_{\text{予測対象命令}} (\text{利得} \cdot \text{頻度} \cdot \text{正解確率})$$

を高めることが重要である．利得と頻度はタスクの定義方法（スケジューリング方法）に依存し，正解確率は予測機に依存する．ただし，利得を有する複数の命令が連続的に発行された場合，この総和が全て直接的に実行時間の短縮に寄与するわけではないため，総実行時間の短縮が上式に比例するわけではないことに注意が必要である．本稿では，個々の値予測の利得という意味に注目してこの利得の定義を用いることにする．

以降の章において，具体的なベンチマークプログラムに対して選択的値予測のシミュレーションを行い，選択的値予測の効果をスケジューリングや利得の観点から考える．

3. シミュレーションモデル

3.1 プログラムモデルと投機的実行モデル

図4に示すようなプログラムフローモデルを考える．丸印は，基本ブロックを意味する．ひとつまたはいくつかの基本ブロックをまとめてタスクを形成し，それを処理要素にマッピングすることをスケジューリングという．本稿では，スケジューリングの違いが全体の実行時間に対してどのような影響を与えるかを調査するために，

- 単純スケジューリング
- ループスケジューリング

と呼ぶ2種類のスケジューリングを考える．

図5にスケジューリングモデルを示す．単純スケジューリングとは，図5(a)に示すようにタスクをひとつの基本ブロックとして，それらを順にPEの番号の若い方から割り当てることにする．一方ループスケジューリングとは，図5(b)に示すように，ループを構成するいくつかの基本ブロックをまとめてタスクを構成し，ひとつの処理要素に割り当てる．

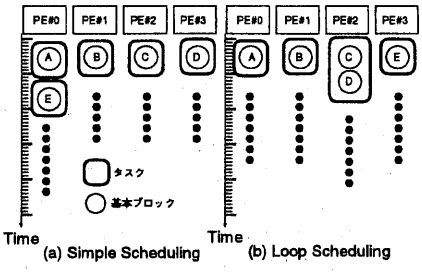

図5 スケジューリング

3.2 値予測方式

現在，値予測方式はいくつか提案[3]されているが，本稿では最も単純な差分を利用した値予測方式を用いることにする．これは，同一プログラムカウンタ(PC)が生成する値を過去の履歴に基づいて予測するものである．差分を利用した値予測方式では，予測値は，

予測値 ＝ 前回の値＋差分；
差分 ＝ 前回の値－前々回の値；

により求める．ここで，前回の値とは，同一PCの命令が前回に生成した値を意味し，差分とは，同一PCの命令が生成した前回の値と，前々回の値の差を意味する．この方式により値を予測すると，次の3種類の系列を予測することが可能となる．

(1) same系列 (例：… 3, 3, 3, 3 …)

(2) stride系列 (例：… 4, 5, 6, 7 …)
(3) 巡回stride系列 (例：… 1,2,3,1,2,3, …)
なお，上の予測方式では，巡回stride系列では，系列の周期の時点でいくつかの予測誤りが発生する．

この予測方式を実現する仕組みとして，図6に示す値予測テーブルを用意する．値予測テーブルでは，PCをタグ情報として，前回の値と差分にアクセスし，その和を上式で示すように予測値として得る．値予測が成功した場合，その値を予測テーブルに書き込む．また，値予測が失敗した場合は，正しい値と正しい値と前回の値の差分を予測テーブルに書き込む．なお，以下の章で本予測方式値予測を行う場合，テーブルの大きさを無限大としてシミュレーションを行う．

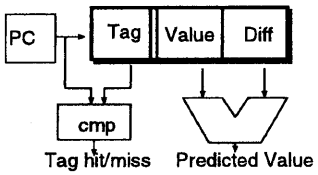

図6 値予測テーブル

3.3 シミュレーション環境

上述の値予測に基づく投機的実行を評価するために，アプリケーションのトレースを解析する評価システムを作成した(図7)．実行トレースを得るためのトレーサとしては，Shade[4]を用いている．Shadeによりsparcプロセッサにおける実行とレースを得ることが可能である．この評価システムは，トレースデータを

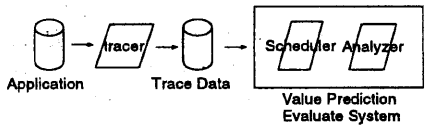

図7 解析ツール

複数の処理要素に割り付けるスケジューラと，命令実行アナライザで構成されている．スケジューラでは，単純スケジューリングもしくはループスケジューリングを行う．スケジューリングを終了したトレースデータに対して，この評価システムでは，

- 依存グラフの作成
- 値予測が必要な命令の候補作成
- 値予測
- 実行時間の評価

を行う．実行時間の評価は，全ての命令が1単位時間で終了するものとし，値予測自体のレイテンシを0，値予測の誤りによるペナルティを2単位時間とした．なお，値予測の検証は値依存が解消された時点で行い，もし値予測が誤っていた場合はその時点からペナルティサイクルを2単位時間課すものとしている．

本稿では，

- SPECint92
- ループプログラム

の2種類のベンチマークプログラムを選んで実験する．ここでループプログラムとは，Livermore Loopの1個のカーネルを模したループが支配的な整数プログラムを人工的に作成した．

4. シミュレーション結果

4.1 SPECint92の結果

SPECint92の各プログラムには，単純スケジューリングのみを行い，選択的値予測を導入して4台の処理要素により投機的実行を1M命令分行った．SPECint92に対して2.2節で述べた基準で選択的値予測を行った場合の予測対象命令と非対象命令の割合を表1にまとめる．

表1 選択的値予測の対象命令と非対象命令 [%]

	com	eqn	esp	gcc	li	sc	Ave.
依存無し	52.4	47.3	52.0	57.8	53.0	48.2	51.8
解決済	38.1	42.0	38.9	32.0	35.4	42.6	38.1
予測対象	9.5	10.7	9.1	10.2	11.6	9.2	10.2
load	15.8	14.5	24.5	22.0	42.6	18.9	23.0
store	31.8	14.7	19.7	14.7	13.7	13.5	18.0
others	52.3	70.8	55.8	63.4	43.6	67.6	58.9

表1において，“依存無し”とは，依存関係がなかった命令（非対象）の割合を表し，“解決済”とは依存はあったが，当該命令が実行されるときまでにその依存が解決されたことを示す（非対象）．これら以外の予測が必要な命令の割合と，その内訳をロード命令，ストア命令および，それ以外の命令の割合として示した．この表からSPECint92を単純スケジューリングした場合，平均で10%程の命令が値予測の対象であり，そのうちの40%程がロードストア命令であることがわかる．

これらの予測対象命令に対して，実際に値予測を行い，投機的実行を4台の処理要素に対して行った場合の実行時間に関する実験結果を図8に示す．図8では，予

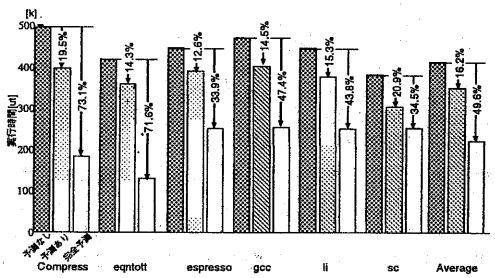

図8 実行時間

測を行わなかった場合と，本稿で述べている予測機を用いた場合，さらに 100% の確率で予測が正解した場合のプログラム実行時間を SPECint92 の各プログラムにおいて評価している．SPECint92 のプログラム全体で平均を取ると，本稿で提案する選択的値予測を行った場合，平均で値予測を行わない場合の実行時間の 16.2% 減となることがわかった．同様に，100% の確率で値予測が成功した場合は 49.6% 減となることがわかった．

また，これらのベンチマークプログラムに対する値予測の正解確率とその命令種別の内訳を図 9に示す．図

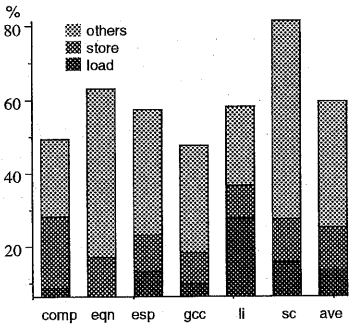

図 9　予測正解確率

9に示すように SPECint92 全体で平均すると，予測確率は 59.5% である．以上のことから，プログラム全体の命令に対して，選択的に 10.2% の命令を 59.5% の確度で予測することにより，全体で 16.2% の実行時間の短縮が図られることになる．

また，個々の値予測がどれほど実行時間の短縮に貢献しているかを調べるために，2.3節に示した利得について調査する．図 10 に SPECint92 の各ベンチマークプログラムにおいて，頻度と利得の関係をを示す．図 10か

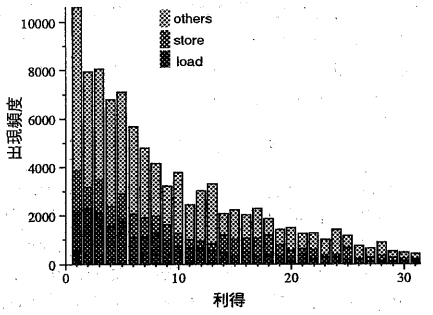

図 10　利得と頻度の関係

らわかるように，利得が大きくなるとその出現頻度は小さくなる．利得が 30 命令までの発生頻度が 98% 以上を占めている．

さらに，それぞれの利得において，どれほどの正解確率が得られているかを調査するために，利得と正解確率の関係を図 11に示す．図 11 は SPECint92 のベンチマークの平均である．図 11からわかるように，利得と

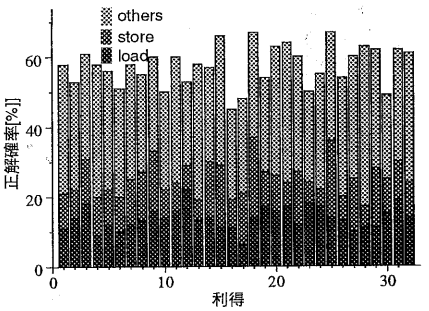

図 11　利得と正解確率の関係

その時の正解確率には相関がない．すなわち，利得が大きいからといって，正解しにくいという訳ではない．

4.2　ループプログラムの結果

2.3節で述べた通り，ループスケジューリングとは，いくつかの基本ブロックで構成されるループ毎に処理要素に割り当てるスケジューリング方法である．図 12に，ループを持つプログラムフローに対する，ループスケジューリングと単純スケジューリングを示す．ループスケジューリングでは，ひとつのタスクが長くなるので，値予測による利得が大きくなり，最終的に実行時間が短縮されると期待される．

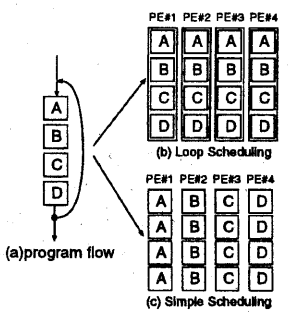

図 12　ループスケジューリングと単純スケジューリング

本稿では，ループスケジューリングの最も単純な例として Livermore Loop の 1 個のカーネルを模した，ループが支配的な整数プログラムを人工的に作成して評価する．このプログラムは，ループ本体の長さがアセンブリコードで約 200 命令で，そのループ本体は平均 50 命令の 4 つの基本ブロックで構成されている．この評価プログラムに対して，ループスケジューリングと単純スケジューリングを施し，値予測を行った．図 13 にループスケジューリングの場合の利得と頻度の関係を，図 14に

単純スケジューリングのそれを示す．それぞれの図に

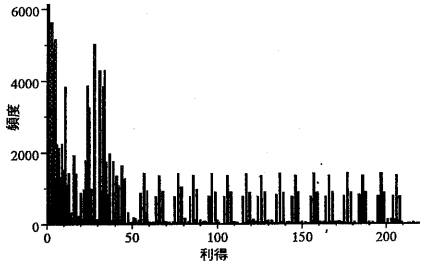

図13　利得と頻度の関係（ループスケジューリング）

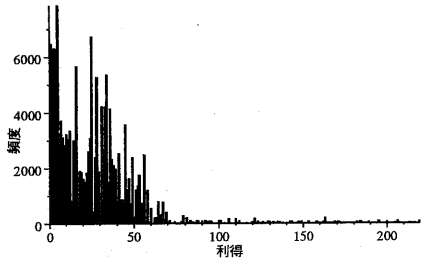

図14　利得と頻度の関係（単純スケジューリング）

示すように，ループスケジューリングの方が利得の大きい値予測が数多く発生していることがわかる．また，本稿ではデータを示さないが，利得の大小と正解確率の間には相関は認められないことがわかった．

これらのスケジューリング方法と実行時間の関係を図15に示す．ループスケジューリングも単純スケジューリングも予測正解確率はどちらもほぼ同じで，約 82.3% と 82.7% であるが，ループスケジューリングでは実行時間が 40.5% 短縮されるのに対して，単純スケジューリングでは 59.8% の短縮を達成している．これは，ここで用いたプログラムがループ間に跨るデータ依存よりも，ループ内に存在するデータ依存の方が支配的であったためだと考えられる．すなわち，単に 2.3節で定義した利得を大きくするという観点だけでスケジューリングしてもトータルの実行時間を必ずしも短縮するわけではないことを意味する．

5. ま と め

本稿では，選択的値予測という概念を導入し，値予測に基づく投機的実行の可能性について述べた．SPECint92 に対して単純スケジューリングを施し，値予測に基づく投機的実行を行った結果，値予測の対象となった命令は全命令のうちの高々 10.2% 程であった．それらの命令に対して，単純な予測機で値予測を行った結果，59.5% の正解確率を達成できた．4 台の処理要

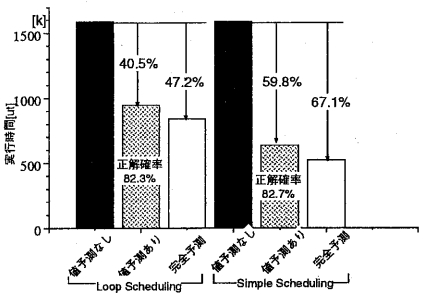

図15　ループスケジューリング

素を用いて投機的実行を行った結果，平均で 16.2% の実行時間の短縮が達成されることが明らかになった．また，個々の値予測に対して，値予測の利得という概念を導入した．利得の大小とその時の正解確率には相関が見られないことが明らかになった．これは，値予測による利得が大きいものでも，必ずしも値予測が正解しにくいというものではないということを示している．

さらに，長いループを含むプログラムに対して，利得が大きくなるようなループスケジューリングと単純スケジューリングを比較した結果，本プログラムでは，ループスケジューリングでは 40.5% の実行時間の短縮であったが，単純スケジューリングでは 59.8% の実行時間の短縮に及んだ．

値予測に基づく投機的実行を効果的に行うためには，

$$\sum_{\text{予測対象命令}} (\text{利得} \cdot \text{頻度} \cdot \text{正解確率})$$

を高めることが重要である．利得と頻度はスケジューリング方法に依存し，正解確率は値予測機の構成に依存する．しかし，トータルの実行時間を短縮させるためには，上式における利得は，2.3節で定義している命令ごとの単純な利得では十分でなく，連続的に発行される複数命令にも対応した「真の利得」とでも言うべき値を考慮しなければならないことが明らかになった．今後は，選択的値予測が有効となるようなスケジューリング方法をコンパイラとともに検討する予定である．

参 考 文 献

1) Mikko H. Lipasti and John Paul Shen: "Exceeding the Dataflow Limit via Value Prediction", In *Proceedings of Micro-29*, Dec. 1996.
2) 山名，佐藤，児玉，坂根，坂井，山口: "並列計算機 EM-4 におけるマクロタスク間投機的実行の分散制御方式"，情報処理学会論文誌，Vol.36,No.7,pp.1578–1588. July 1995.
3) Yiannakis Sazeides and James E. Smith: "The Predictablility of Data Values", In *Proceedings of Micro-30*, Dec. 1997.
4) http://sw.sun.com/shade/