Vol.2010-ARC-188 No.7
2010/3/1

値予測機構を有するマルチプロセッサに適したスレッド粒度の検討

伊波　健[†1]　中野秀洋[†1]　宮内新[†1]

従来研究において，命令レベル並列性を抽出する値予測とスレッドレベル並列性を抽出するマルチスレッド実行を用いて，さらなる性能向上を目指した値予測機構を有するマルチプロセッサが提案されている．しかし，その性能評価は各スレッド単体に対する基本的な評価であり，各スレッドの VHT 上の競合や各プロセッサへの割当てなどに関する詳しい調査が行われていない．本論文では，同プロセッサに関して，スレッド粒度と最大性能向上に必要な VHT サイズの関係性から，同プロセッサに適したスレッド割当てについて議論する．

Investigation of thread grain size suitaeal for a Multi-processor with a value predictor

Ken Inami,[†1] Hidehiro Nakano[†1]
and Arata Miyauchi[†1]

The conventional research has proposed Multi-processor with value predictor improving instruction level parallelism(ILP) and thread level parallelism(TLP). However, in the conventional research, the basic performances of the multi-processor have been evaluated only in the cases where for single each thread is execued. So, competitions on VHT and an allocation scheme to each processor have not been considered so far. This paper discusses suitable allocations of each thread in this multi-processor from the viewpoints of relationships between thread grain size and performances.

†1 東京都市大学
Tokyo City University

1. はじめに

現在，マルチスレッドプロセッサは組み込みシステムからワークステーション，各種サーバに至る多くのシステムで採用されている．このようなアーキテクチャ上では複数のスレッドを同時実行することにより性能向上を図っている．また，さらなる性能向上のために，様々な並列化コンパイラや，スレッドスケジューリング技術が数多く提案されている[1)]．これらは，プログラム中のループやサブルーチン，基本ブロック間の並列性であるスレッドレベル並列性に注目したものである．

一方，命令レベル並列性を抽出する方法として，値予測[2)] が提案されている．プログラムにおいて、命令レベル並列性を制限する大きな要因となっているのが、制御依存とデータ依存である．このうち，制御依存は分岐予測により解消可能である．データ依存は，出力依存，逆依存，そして真の依存に分類されるが，このうち出力依存と逆依存についてはレジスタリネーミングにより解消可能である．残る真の依存については解消は困難とされてきたが，値予測はこの依存関係を解消する手法として近年注目されている．

値予測は，演算の結果生成されるデータ値を予測することで処理を進める投機処理技術である．値予測は，命令が繰返し実行される際に，過去に生成された値の履歴から予測を行う．したがって命令毎に生成された値の履歴を保持するテーブル (VHT: Value History Table) が必要となる．ハードウェア量の制限から，VHT のサイズには限界があるが，プログラムの規模やループ構造によっては，VHT 上の競合による容量性ミスが増加してしまうという問題点がある．

従来の研究では，それらの問題を解決し，前述した 2 つの並列性を抽出することで更なる性能向上を図るため，マルチプロセッサに値予測機構を適用する手法の提案と評価が行われている[3)]．しかし，その性能評価は各スレッド単体のみしか行われておらず，各スレッドの VHT 上での競合や各プロセッサへのスレッド割当てなどに関する詳しい調査は行われていない．そこで，本研究ではスレッド粒度及び値予測可能命令数と最大性能向上に必要な VHT サイズとの関係性を調査し、値予測に適したスレッド割当てについて議論する．

2. 値予測

2.1 値予測機構

値予測[2)] には種々の方式が提案されている．代表的な値予測機構として，最終値予測，ストライド値予測がある．

© 2010 Information Processing Society of Japan

最終値予測は，同一 PC (Program Counter) を持つ命令について，最も近い過去に得られた値をそのまま予測値とする手法である．つまり，同じ演算結果が繰り返されると予測する．最終値予測機構では一般に，予測ミスの発生回数削減のために，動的に変化するカウンタからなる分類テーブル (CT: Classification Table) を用いる．この方法では，予測が正解していればカウンタを 1 つ増加させ，予測ミスであれば 0 に初期化する．そして，このカウンタが閾値に達するまで予測値を行わない．

ストライド値予測は，最も近い過去に得られた値に，過去 2 回の演算結果の差分 (ストライド値) を加えて予測値を計算する手法である．ストライド値予測機構では一般に，予測ミス発生回数削減のために，$Init$，$Transient$，$Steady$ の 3 つの状態をもつ CT を用いる．この方法では，命令の実行のたびに状態を変化させ，ストライド値が安定するまで予測を行わない．

2.2 値履歴テーブル

値予測を行うためには，その命令に関する履歴，予測状態の情報が必要となる．最終値予測では最も近い過去に得られた値とカウンタ値，ストライド予測では最も近い過去に得られた値，ストライド値，状態の情報が必要である．これら，値予測に必要な履歴情報は VHT に格納され保持される．VHT へのアクセスは，各命令のアドレスをもとに，機構が単純・高速なダイレクトマッピング方式によって実現される．

2.3 予測ミスと容量制ミス

値予測による性能向上を制限する大きな要因として，予測ミスと容量性ミスがある．

予測ミスとは，予測した値が実際に命令が実行されて得られた値とは異なることである．この場合，値予測によって得られた値は誤った値であるので，予測値を使用した後続命令の実行をやり直さなければならない．このように，予測ミスが発生すると性能向上どころか，値予測を行わない場合よりも余計に実行サイクルを消費してしまう．

容量性ミスとは，予測を行う命令の履歴が存在しないことである．これは命令の初回実行時以外にも，VHT 上の競合により発生する場合がある．値予測の対象となる命令すべての履歴を格納できるような大きさの VHT を用意することは現実的ではない．そこで，VHT の 1 つのエントリを複数の命令が利用することになる．このとき，それらの命令の実行順序によっては同一エントリで競合が起きる場合がある．競合が起きると，過去の履歴を参照できなくなるため，値予測は行われない．従って，VHT 上の同一エントリを使用する命令が高い頻度で入れ替わる場合，VHT の更新だけが繰り返され，値予測による性能向上が得られないという問題が発生する．

2.4 値予測命令選定

値予測においては値を生成する全命令を予測対象の候補とすると，予測ミスによるペナルティが非常に大きくなる．また容量性ミスも頻繁に発生する．この問題を解決するために，プログラムを実行する前にソフトウェアを用いて値予測対象とする命令を選定する値予測命令選定法が考案されている．

予測ミスに対しては，プロファイリング情報を用いて予測正解率の高い命令のみを予測対象とする方法が，最も単純であり効果的である[4]．

容量性ミスに対しては，予測正解率，命令の依存関係，命令の種類等の情報を基に，予測正解時の利得が小さい命令を除外することで，予測対象命令数を削減する方法が提案されている[4]．また，VHT のエントリ数を考慮した命令選定法も提案されている[5][6]．

しかし，容量性ミスについては，プログラムの規模やループ構造への依存が大きく，プログラムによっては容量性ミスによる性能低下を解消しきれないという問題点がある[5]．

3. マルチスレッドプロセッサ

従来のプロセッサは命令レベル並列性を利用し，高速化を図ってきたが，近年はスレッドレベル並列性を利用したマルチスレッドプロセッサが注目されている．マルチスレッドプロセッサでは，1 つのプログラムが規模や構造を基にスレッドとして分割され，それらスレッド間にあるスレッドレベル並列性が抽出される．これによって各スレッドが同時に並列実行される．マルチスレッドプロセッサには，SMT プロセッサ，マルチプロセッサ，マルチコアプロセッサ等がある．

SMT プロセッサは，単一のプロセッサ上で動作していないプロセッサ内部の実行ユニットを利用して複数スレッドを同時実行する．一方，マルチプロセッサ，マルチコアプロセッサは，実装された複数の処理要素によって複数スレッドを同時実行する．

スレッドは，プログラム中で明示的に指定されることもあるが，自動的に抽出することも可能である．こうしたスレッド抽出は並列化コンパイラの機能として実装される．これらのコンパイラでは，さらなるスレッドレベル並列性をを引き出すための最適化も行われる．

初期においては，ループ並列化コンパイラが主流であった．これは，ループの各イテレーション間に存在する並列性に注目したものである．これらループ並列化技術が成熟してくると，他のスレッド粒度での並列性も注目されてきた．

例えば，アドバンスド並列化コンパイラプロジェクトの OSCAR マルチグレイン並列化コンパイラ[1] は，中粒度であるループ並列性の他に，粗粒度並列性，細粒度並列性を階層的

© 2010 Information Processing Society of Japan

に組み合わせて使用することで，プログラム全体から並列性を抽出することができる．ここで，粗粒度並列性とは，サブルーチン，ループ，基本ブロック間の並列性である．また，細粒度並列性とは，複数命令レベルの並列性である．

4. 値予測機構を有するマルチプロセッサ

4.1 本機構の特徴

従来研究において，値予測機構を有するマルチプロセッサが提案されており[3)]，基本的な性能評価が行われている．この研究では，スレッドレベル並列性と命令レベル並列性をともに抽出することでプログラム全体の性能向上を図ることを目的としている．

この手法の特徴には以下の 3 つがある．

- 分散 VHT 方式
- プロファイリングによる予測ミスへの対応
- 中粒度以上のスレッド粒度を対象

VHT をマルチスレッドプロセッサに搭載する場合，その実装方式として，全ての処理要素 (PE: Processor Element) で共通して持つ共有 VHT 方式と，各 PE が別々に持つ分散 VHT 方式が考えられる．

共有 VHT 方式では，値予測の正解数や，容量性ミスの発生回数などについては，従来と同じ結果が得られると考えられる．しかし，複数 PE から同時にアクセスされることを考慮すると，マルチポート化による面積増大や，アクセスレイテンシの増加が考えられる．

分散 VHT 方式では，各 PE 毎に VHT を持つため，アクセス面では問題ない．また，各 PE がスレッド単位で処理を行うため，値予測の対象もスレッド単位となる．プログラムの規模や構造によってスレッド分割を行えば，従来の値予測機構の問題点であったプログラムの依存による容量性ミスの発生を抑えることができると考えられる．

また，従来研究では，VHT のサイズを PE ごとに異なるものとするヘテロジニアスな構造を持つマルチプロセッサである方が性能とハードウェアコストの面から，実用に適していると結論づけている．これは，値予測の効果が得られないスレッドが存在するためである．さらに，スレッド構造によって，それぞれ値予測による効果を得るための VHT サイズは異なると考えられる．そのため，各 PE に対して VHT をどのように割当てを行うのかという検討も必要である．

予測ミスが頻繁に発生してしまうと性能低下につながる．よって，値予測命令選定によるプロファイリング情報を用いて予測正解率の高い命令のみを予測することで対応する．この手法は，あらかじめプログラムを実行させ，各命令ごとの予測正解率をプロファイリング情報として得る．そしてその情報から，予測正解率の高い命令のみを予測対象とするものである．これは，プロファイリングに用いる入力と実際の実行に用いる入力が異なっていても，似たような予測正解率を得られるというプログラムの性質を利用したものである．

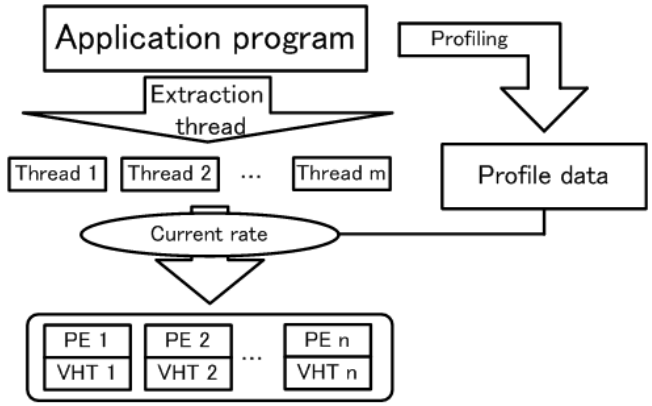

図 1 値予測機構を有するマルチプロセッサ
Fig. 1 Multi-processor with value predictor

次に，スレッド粒度に関して説明する．値予測は命令が繰り返し実行されることで性能向上を得る．よって，対象となるスレッドは繰り返しを含むものでなくてはならない．そのため，ループの各イタレションをスレッドとする中粒度スレッド，また，サブルーチン，ループ単位をスレッドとする粗粒度スレッドを利用する．

図 1 に値予測機構を有するマルチプロセッサの概要を示す．

4.2 従来研究の課題

従来研究では，単体の各スレッドに対する基本的な性能を評価している．それによって，値予測機構に適したスレッドの調査を行っている．しかし，実際には，1 つの PE 上では複数のスレッドが処理される．そのため，スレッド割当てやプログラム全体の性能評価が十分になされていないことが問題点として挙げられる．また，各 PE にどの程度の VHT サイズを用意すべきかなどの検討がなされていない．

4.3 研 究 目 的

本研究では，従来研究の課題を検討するため，スレッドの粒度やそのスレッドに含まれる

© 2010 Information Processing Society of Japan

値予測命令数に対して必要な VHT サイズを，ベンチマークプログラムを用いたシミュレーションによって詳しく調査する．そして，各スレッドの性能向上に対して必要な VHT サイズをスレッド粒度などによって推測できるかを検討する．

5. スレッド調査

本研究のスレッドの調査では，各スレッドが値予測による最大性能向上が得るための VHT サイズを求め，その VHT サイズとスレッドの粒度，そのスレッドに含まれる値予測命令数の関係性を明らかにする．その調査の手順を図 2，図 3 に示す．

プロファイリングにより値予測に効果のある命令アドレスが得られる．これを用いて，値予測による効果がないスレッドを選定できる．今回求める対象は最大性能向上に必要な VHT サイズであるため，値予測の効果がないスレッドを除外し，プロファイリング情報から値予測に効果が期待できるスレッドを抽出する．具体的には，値予測正解率が 95%以上の命令アドレスを含むスレッドを抽出する．

そして，抽出した各スレッドをシングルプロセッサ上で処理を行い，必要な VHT サイズを得る．得られた最大性能向上に必要な VHT サイズとスレッド粒度及び値予測命令数との関係性を評価する．

6. 実　　験

実験では，スレッド粒度及び値予測命令数と最大性能向上に必要な VHT サイズとの関係性を調べる．

6.1 実験環境

実験には，スーパースカラシミュレータの SimpleScalar Ver.2.0[7] に値予測機構を追加した Data Value Predictor Simulator[8] を，スレッド単位での値予測が可能なように改造したものを使用した．値予測法としては，ストライド値予測手法を用いた．

実験に使用したプログラムは，SPECint95 ベンチマークの gcc，go の 2 種類である．スレッド抽出は，これらの実行バイナリをディスアセンブリし，制御フロー解析を行うことによって，サブルーチン単位のスレッドの中で繰り返しが多い順に 90 スレッド用意した．表 1 に，値予測の効果があるスレッドと効果のないスレッドの内訳を示す．

6.2 実験結果

図 4 から図 7 に gcc，go のベンチマークの実験結果をそれぞれ示す．図 4，図 6 の縦軸は最大性能向上に必要な VHT サイズ，横軸はスレッドの粒度である．図 5，図 7 の縦軸は

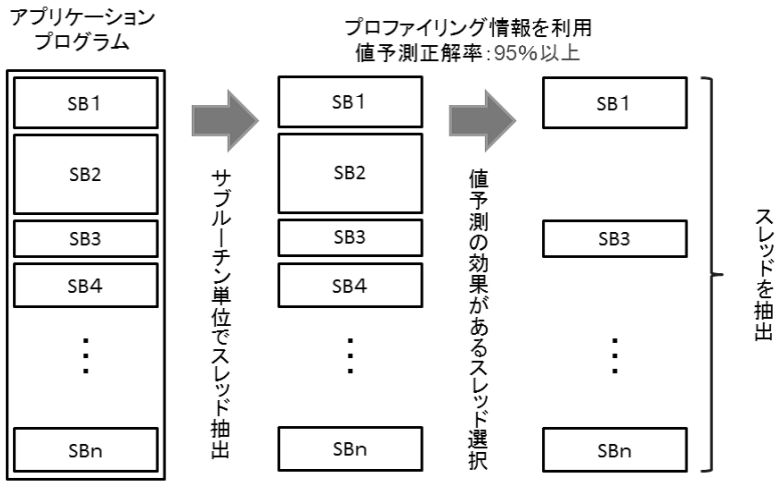

図 2 スレッドの抽出
Fig. 2 Extracting thread

表 1 各ベンチマークのスレッド数
Table 1 The thread num by some benchmark

ベンチマーク	gcc	go
総スレッド	2025	480
値予測の効果があるスレッド	588	315
値予測の効果がないスレッド	1437	165

最大性能向上に必要な VHT サイズ，横軸はスレッドに含まれる値予測の効果がある命令数である．

両方のベンチマークの実験結果からスレッドの粒度や値予測命令数が増加すれば性能向上を得るための VHT サイズは増加することが分かる．また，値予測命令数を大きく越える VHT サイズを必要としていることが分かる．これは，VHT がダイレクトマッピングであるため，VHT の競合が起きるためである．スレッド粒度と VHT サイズの関係をみると，VHT サイズを 2^n としたとき，基本的に「2^{n-1} < スレッド粒度 < 2^n」を満たす VHT が最大の性能向上を得るために必要な最小の VHT サイズであることが分かる．

さらに，従来研究[5] や今回の実験結果から，性能向上に必要な VHT サイズがスレッドに

© 2010 Information Processing Society of Japan

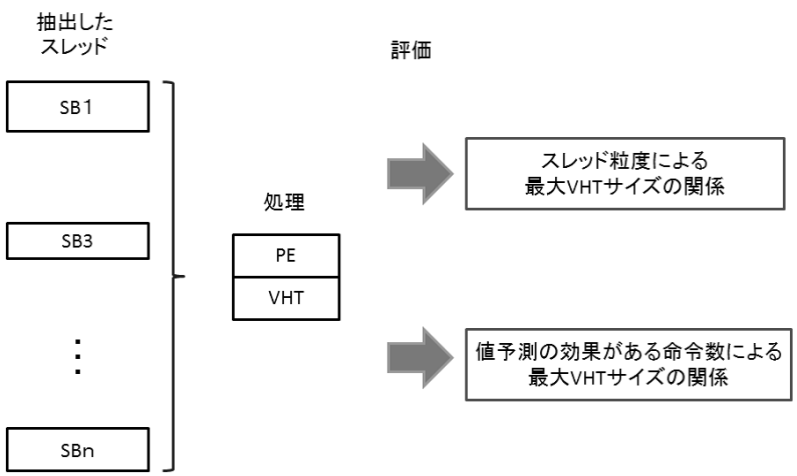

図 3 各スレッドの評価
Fig. 3 Method of eval some threads

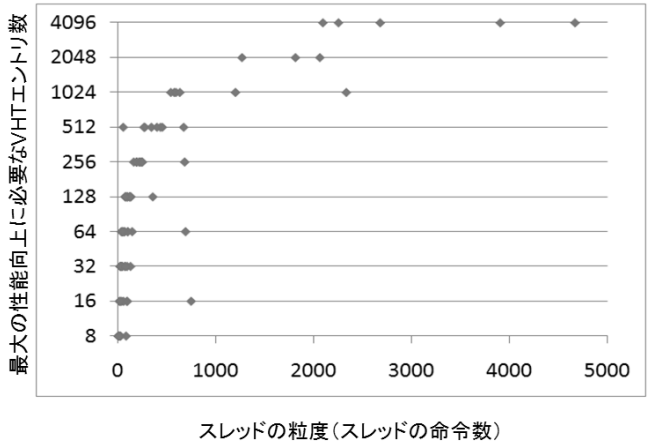

図 4 gcc: スレッド粒度と VHT サイズの関係
Fig. 4 gcc: Relationship thread grain and VHT

よって様々であることが確認できる．例えば，大きい VHT サイズを必要とするスレッドとそうでないスレッドがあるとき，各 PE に VHT を均等に割り振ることはハードウェアコストの観点から効率的ではない．値予測の効果が期待できないスレッドについては実行する PE が VHT をもつ必要がない．これらのことから，VHT サイズが PE 毎に異なるヘテロジニアスな構造をしたマルチプロセッサが適しているといえる．

以上より，プロファイリング時に値予測に効果があるスレッドの粒度をみることによって，そのスレッド粒度よりも大きい VHT を持つ PE に割当てることで，そのスレッドにおいて最大の性能向上を得ることができると考えられる．

6.3 考　　察

実験の結果から，プロファイリング情報を用いることで，値予測の効果があるスレッドを抽出でき，更にスレッド粒度の情報を用いることで各スレッドに必要な VHT サイズの推測が可能であるが確認できた．スレッド粒度よりも大きい VHT を持つ PE に割当てることでそのスレッドにおいて最大の性能向上を得ることができる．

従来研究で，複数のスレッドを 1 つの PE 上で処理したときのスレッドの VHT の競合に関して，評価と検討を行っている[9]．また，各スレッドを性能向上に必要な VHT サイズが大きいもの，小さいもの，値予測の効果がないというスレッドに分けたとき，大きい VHT サイズを必要とするスレッド同士で処理を行うと VHT で競合が発生し，性能が低下することを確認している．このため，必要とする VHT サイズが大きいスレッド同士は，異なる PE で処理を行うべきであると結論づけている．

図 4 から図 7 における実験の結果から，スレッドの粒度によって必要な VHT サイズは推測できる．そこで，値予測の効果のあるスレッドの中でスレッドの粒度が大きいスレッド同士は，スレッド粒度よりも大きい VHT サイズを持つ異なる PE にそれぞれ割り振ることで VHT の競合を防ぐことが可能となり，効果的な割当てを行えると考えられる．

7. おわりに

本研究では，スレッド粒度や値予測命令数によって必要となる VHT サイズを検討した．実験の結果から，プロファイリング情報に基づきスレッド粒度よりも大きい VHT サイズに割当てる方が値予測の効果が見込めると考えられる．

今後の課題としては，プロファイル情報を利用したコンパイラの開発などが考えられる．また，実験を行ったベンチマークが gcc，go のみであり，他のベンチマークでも同様の結果が得られるかを検討する必要がある．

© 2010 Information Processing Society of Japan

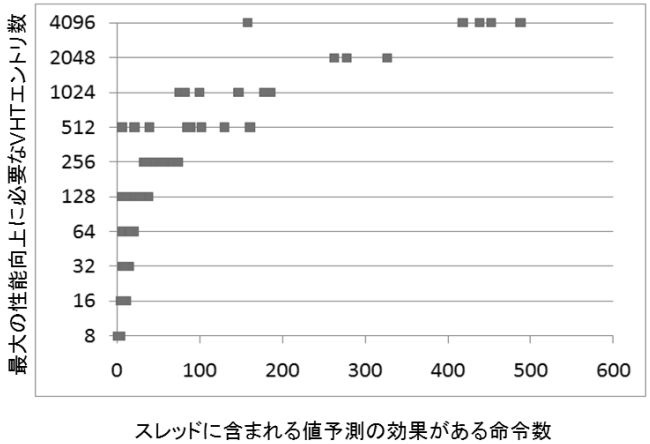

図 5 gcc: 値予測命令数と VHT サイズの関係
Fig. 5 gcc: num of instruction value predictor and VHT

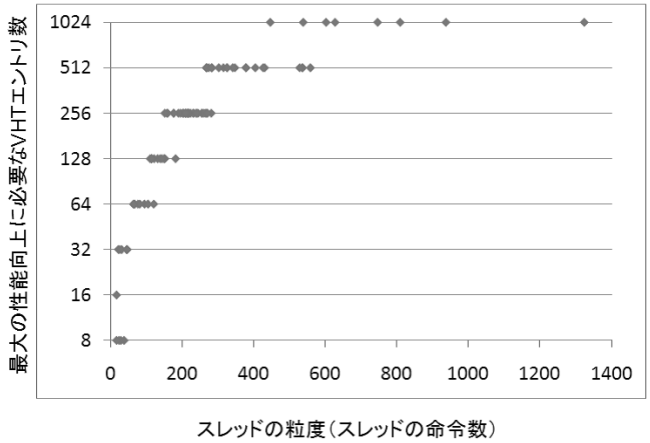

図 6 go: スレッド粒度と VHT サイズの関係
Fig. 6 gcc: Relationship thread grain and VHT

参 考 文 献

1) 小幡元樹他:「マルチグレイン並列処理のための階層的並列性制御法」，情処論，Vol.44 No.4，pp.1044-1055(2003)
2) 斎藤史子他:「投機的実行に関する最新技術動向」，情処研報論，Vol.44，No6，June 2003
3) 菅野雅之他:「マルチスレッドプロセッサに適した値予測機構の提案」，電子情報通信学会総合大会，D-6-5，2007
4) 飯塚大介他:「値予測の軽量効率化方式の提案と評価」，情処論，Vol.44，No6，June 2003
5) 菅野雅之他:「値履歴テーブルのエントリ数に適した値予測命令数の検討と評価」，信学 2005 年総合大会，D-6-3，p51，2005，3
6) 藤原亮人他:「値履歴テーブルのエントリ数に対応可能な値予測命令選定法」，情処第 67 回全国大会，1ZB-4，Mar.2005
7) SimpleScalar LLC,http://www.simplescalar.com/
8) Sang-Jeong Lee,Soonchunhyang University,Asan, Chungnam,Korea.http://sjlee.sch.ac.kr/
9) 伊波健他:「マルチスレッドプロセッサに適したスレッド割当ての検討」，電子情報通信学会ソサイエティ大会，A-8-3，Sep.2009

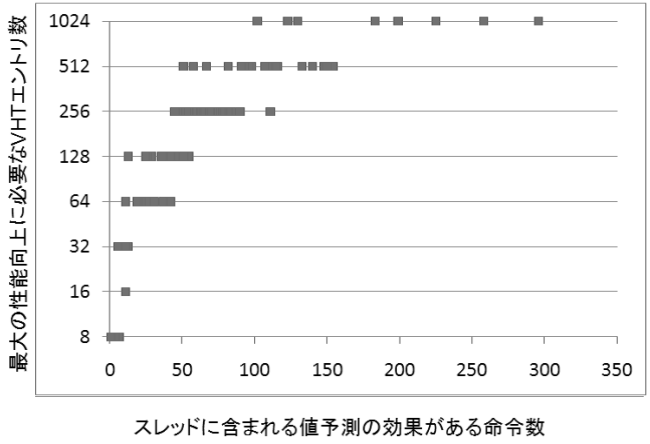

図 7 go: 値予測命令数と VHT サイズの関係
Fig. 7 go: num of instruction value predictor and VHT

© 2010 Information Processing Society of Japan