

コンパイラ最適化レベルのデータ投機実行に与える影響

濱野 彰彦¹ 杉谷 樹一² 佐藤 寿倫^{3,4} 有田 五次郎³

¹ 有限会社ハート・アット・ワーク ² 富士通九州デジタルテクノロジー株式会社

九州工業大学³ 情報工学部 知能情報工学科 ⁴ マイクロ化総合技術センター

命令の演算結果を予測し、その予測値を使用してデータ依存関係を投機的に解消することで、より大きな命令レベルの並列性を抽出しようという試みがなされている。これまで、より大きな予測可能性や予測精度を追求した研究がたくさん行われてきた。しかし、われわれの知る限りにおいては、コンパイラによる最適化とデータ値予測可能性との間の関係を調査した報告は無い。そこで本稿では、コンパイラの最適化レベルを変化させてコンパイルされたバイナリを用いて、これらの間の関係を調査する。詳細なシミュレーションの結果、コンパイラの最適化レベルを変化させても予測可能性には大きな影響は現われないことが確認された。

Influence of Optimization Levels on Data Speculation

AKIHIKO HAMANO

KIICHI SUGITANI

TOSHINORI SATO

ITSUJIRO ARITA

The practice of speculation in resolving data dependences based on value prediction has been studied as a means of extracting more instruction level parallelism. There are many studies on value prediction mechanisms with high predictabilities. However, to the best of our knowledge, the relationship between compiler optimization level and value predictability has not been investigated. In this paper we evaluate value predictability of several binaries which are compiled with different optimization levels. Detailed simulations reveal that value prediction is still effective for highly optimized binaries.

1. はじめに

現在主流のマイクロプロセッサは性能向上を命令レベルの並列性 (ILP: instruction level parallelism) の抽出に頼っている。しかしながら、ILP は命令間の依存関係によって制限される。それらは、制御依存、資源競合、データ依存の 3 つに分類できる。制御依存と資源競合を取り除く方法には既に多くの研究があるが、これまでデータ依存は ILP を制限する深刻なボトルネックと見做されてきた。近年、より多くの ILP を抽出するためにデータ依存を投機的に解消する方式が注目されている^{5),6)}。ある命令の結果は値予測器を用いて予測され、この命令と依存関係にある命令を同時に実行できる。したがって、ILP を積極的に抽出できるわけである。

しかし現在のところ、データ値予測器に関する検討は主に予測方式に関してである。われわれの知る限りにおいては、コンパイラによる最適化とデータ値予測可能性との間の関係を調査した報告は無い。もしコンパイラが十分最適化を施し、データ値予測に基づいたデータ投機実行による性能向上の余地が無いとすれば、データ投機実行は無意味である。そこで本稿では、コンパイラの最適化レベルを変化させてコンパイルされたバイナリを用いて、これらの間の関係を調査する。

以下に本稿の構成を示す。次節で関連研究をまとめる。3 節では評価環境を説明する。4 節でシミュレーションにより評価する。最後に 5 節で全体のまとめを行なう。

2. 関連研究

値予測に基づいたデータ投機実行^{5),6)} は、命令間の依存関係を緩和し ILP を抽出する新たなパラダイムとして注目されている。これまでに様々な予測器が提案されているが、比較的予測精度の高い 2 レベル型予測器¹³⁾、ハイブリッド予測器¹³⁾、そしてコンテキスト型予測器¹⁰⁾ はハードウェア量が非常に大きくなるという問題がある。

Morancho ら⁷⁾、Rychlik ら⁹⁾、Del Pino ら³⁾ や Calder ら²⁾ は値予測器のハードウェア量を削減することを検討している。Morancho ら⁷⁾、Rychlik ら⁹⁾ と Del Pino ら³⁾ の方式では、演算結果の予測容易性に基づいて命令を分類し、予測容易な命令はハードウェア量の小さな予測器を使用し、予測困難な命令だけ上述のハードウェア量の大きな予測器を使用する。つまり、ハイブリッド値予測器のハードウェア量削減に着目している。しかし、構成要素となる各値予測器のハードウェア量は削減できてはいない。Calder ら²⁾ は予測可能性に着目して、値予測表に保持される命令を選択している。データ投機実行に有効な命令だけを選択することで値予測表の利用効率を改善し、その結果エントリ数の削減を達成している。一方、Fu ら⁴⁾ と Tullsen ら¹²⁾ は、コンパイラの支援によって予測器のハードウェアを完全に取り除くことを検討している。

コンパイラによる最適化とデータ値予測可能性との間の関係については、既に文献 15), 16) において報告した。

表 1 実行命令数 (GCC)

program	-00		-01		-02	
099.go	268,225,895	(223,777,492)	146,032,803	(115,999,78)	134,766,291	(104,239,257)
124.m88ksim	215,027,498	(155,551,958)	124,461,169	(86,167,274)	119,705,428	(81,391,241)
126.gcc	146,724,176	(100,785,825)	105,315,933	(69,649,020)	103,357,196	(67,359,249)
129.compress	77,092,827	(54,636,345)	48,821,478	(32,717,148)	47,719,938	(31,615,607)
130.li	307,778,502	(185,680,534)	208,780,589	(123,022,538)	206,414,110	(121,247,791)
132.jpeg	18,082,420	(14,009,018)	8,831,646	(6,490,266)	8,606,970	(6,255,132)
134.perl	12,166,065	(7,775,114)	10,641,524	(6,611,315)	10,645,288	(6,607,758)

表 2 実行命令数 (MIRV)

program	-00		-01		-02	
099.go	179,701,392	(142,959,132)	131,900,827	(98,921,730)	132,506,520	(99,186,262)
124.m88ksim	184,814,315	(130,030,485)	122,977,073	(82,888,272)	123,766,329	(85,244,701)
126.gcc	140,356,442	(96,215,914)	115,904,906	(75,804,673)	115,334,584	(86,961,482)
129.compress	69,108,399	(48,019,527)	49,210,033	(32,525,461)	47,700,633	(31,587,802)
130.li	270,793,655	(161,858,905)	207,709,666	(121,168,612)	207,705,881	(121,167,937)
132.jpeg	12,045,397	(8,705,739)	9,751,916	(7,033,935)	9,995,242	(7,282,418)
134.perl	12,574,740	(8,233,448)	10,489,957	(6,644,109)	10,510,075	(6,660,903)

本稿ではさらに一歩進めて、コンパイラによる最適化とプロセッサ性能との間の関係を調査する。

3. 評価方法

本節では、シミュレーションに用いたプロセッサのモデル、ベンチマークプログラムとコンパイラを紹介し、提案手法の評価環境について説明する。

3.1 プロセッサモデル

評価には 2 種類のプロセッサモデルを用いる。データ予測可能性は命令レベルの機能シミュレータを用いて評価する。一方、プロセッサ性能の評価にはサイクルレベルのタイミングシミュレータを用いる。どちらのシミュレータも SimpleScalar ツールセット (ver.3.0a)¹⁾ を用いて作成されている。

性能評価に用いられるプロセッサモデルはレジスタ更新ユニット (RUU: register update unit)¹¹⁾ に基づいて動的命令スケジューリングを行なうスーパースカラプロセッサであり、以下の構成となっている。RUU の容量は 128 エントリである。命令フェッチ幅と命令発行幅は 8 命令であり、8 つの機能ユニットは完全に対象であらゆるタイプの命令を実行できる。各演算のレイテンシは乗算 (4 サイクル) と除算 (12 サイクル) を除いて全て 1 サイクルである。データキャッシュは 4 ポートのノンブロッキングキャッシュで、容量 128KB でブロックサイズ 32B の 2 ウエイセットアソシアティブで構成される。データキャッシュのレイテンシはデータアドレス計算の他に、ヒット時 1 サイクル、ミス時 6 サイクルである。また全てのメモリアクセス命令は、先行するストア命令が完了していなければ実行できない。命令キャッシュは容量 128KB で、ブロックサイズ 32B の 2 ウエイセットアソシアティブキャッシュである。2 次キャッシュはデータと命令で共有されており、容量 8MB でブロックサイズ 64B のダイレクトマップキャッシュである。2 次キャッシュミス時のレイテンシは、最初のワードが得られるまでが 18 サイク

ルで、後続のワードのアクセスにはそれぞれ 2 サイクルを必要とする。分岐先アドレスの予測にはエントリ数 1K で 4 ウエイセットアソシアティブの分岐先バッファとエントリ数 8 のリターンアドレススタックを、分岐方向の予測にはエントリ数 4K の gshare タイプ 2 レベル適応型分岐予測器を用いた。これらの分岐予測器の更新は命令のリタイア時に行なわれる。

本稿では、文献 6) の最終値型予測器と文献 13) のハイブリッド予測器を用いて評価する。どちらもダイレクトマップ方式で構成する。前者はシンプルなデータ値予測器の代表として、後者は複雑な予測器の代表として採用した。ハイブリッド予測器はストライド型予測器と 2 レベル型予測器から構成されている。値予測に失敗した場合には、正しい実行を保証するために、間違った予測値を用いて投機的に実行された命令を、正しいオペランドで再実行する必要がある。このための回復機構には、投機に失敗した命令とデータ依存の関係にある命令だけを選択的に再実行する方法を選択した。再実行する時には、予測値と正しい値とを比較するために必要な 1 サイクルの比較ステージがペナルティとなる。

3.2 ベンチマークプログラム

SPEC95 CINT ベンチマークプログラムを用いて評価する。それぞれのプログラムの入力セットは現実的な時間でシミュレーションが終了するように調整してあり、プログラムの終了まで実行する。これらのプログラムは MIPS アーキテクチャを拡張した SimpleScalar/PISA アーキテクチャ¹⁾ をターゲットとしてコンパイルされている。

3.3 コンパイラ

本研究で用いるバイナリはミシガン大学から配付されているものである。それらは GNU GCC (version 2.7.2.3) と MIRV コンパイラ⁸⁾ でコンパイルされている。各コンパイラで 3 つの最適化レベル -00, -01, -02 を評価する。表 1 と表 2 に全実行命令数と予測対象命令数 (カッコ内) を示す。ここでの命令数は命令レベルの機能シミュレー

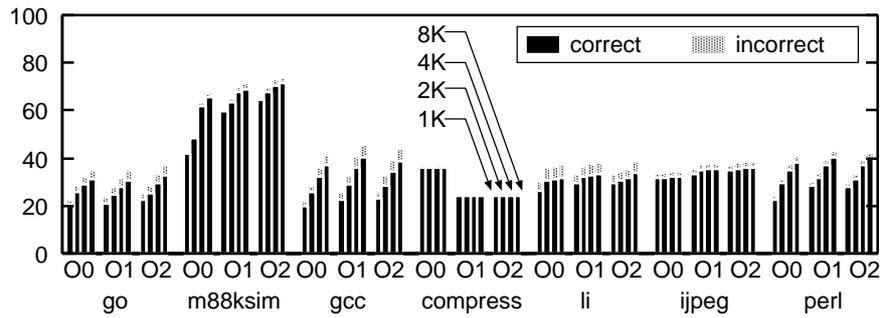


図1 %予測可能性 (GCC/最終値)

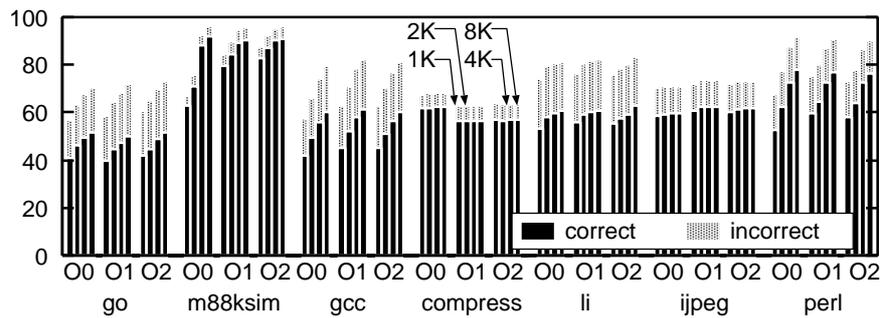


図2 %予測可能性 (GCC/ハイブリッド)

タ用いて得られたものなので、最終値型予測器とハイブリッド予測器で同じ値になることに注意されたい。一般に最適化レベル-00では、MIRVの方がGCCよりも実行命令数が少ない。この理由は、この最適化レベルにおいてMIRVはグラフ彩色を用いている⁸⁾がGCCは用いていないことによると考えられる。他の最適化レベルでは、二つのコンパイラはほぼ同じ性能を示していることが確認できる。

4. シミュレーション結果

本節でシミュレーション結果を紹介する。まず、予測可能性 (predictability)、予測率 (prediction coverage)、そして予測精度 (prediction accuracy) を評価する。これらの指標は以下のとおり定義される。予測可能性とは、予測対象となる全ての命令の中で、予測の成否に関係なく予測可能だった命令の割合を示している。予測率は、予測対象命令の中で、正しく予測できた命令だけの割合を示している。最後に予測精度は、予測可能だった命令の中で、正しく予測された命令の割合を示している。したがってこれら3つの指標の間には以下の関係が成り立つ。

$$(\text{予測率}) = (\text{予測可能性}) * (\text{予測精度})$$

これらの評価に続いて、データ投機実行を用いた際のプロセッサ性能を評価する。

4.1 予測可能性

図1はGCC生成のバイナリに対して最終値型予測器を試した結果である。横軸はプログラム名と最適化レベルで縦軸は予測可能性 (%) である。各グラフは二つの部分に分けられており、予測可能性と予測率をまとめてい

る。下部 (図中黒色) は正しく予測できた割合を示しており、上部 (図中灰色) は間違って予測された割合を示している。残りの部分は、予測しなかった割合である。したがって、グラフの下部が予測率を、上部と下部の和が予測可能性にあたる。各グループに対する4つのグラフは左から順に、予測表の容量が1024エントリ、2048エントリ、4096エントリ、8192エントリの時の結果である。この図から以下のことが観察できる。まず129.compressにおいては、最適化レベルが-00から-01に上がると、予測可能性も予測率も著しく低下していることが判る。このことは高い最適化がデータ値予測を不要にしていることの例に思われる。第2に、上記の結果とは逆に残りのプログラムにおいては、最適化レベルが上がると、予測可能性と予測率が若干向上していることが見られる。このことから、129.compressは特殊なケースであると確認できる。したがって、最終値型予測器は最適化レベルが上がっても有効であると言える。

図2はGCCバイナリにハイブリッド予測器を適応した場合の結果である。図の見方は図1と同じである。最終値型予測器での観測と似た傾向が見られる。しかし、二つのプログラムにおいて違いが観察される。129.compressにおいては、最適化レベルが-00から-01に上がった時の変化が小さい。また134.perlにおいても、最適化レベルが上がった時に予測可能性と予測率の低下が見られる。しかし、このことから最終値型予測器と異なった結論が導き出されるわけではない。已然として残りのプログラムでは、最適化レベルが上がると、予測可能性と予測率が若干向上している。つまり、ハイブリッド予測器も最適化レベルが上がっても有効であると言える。

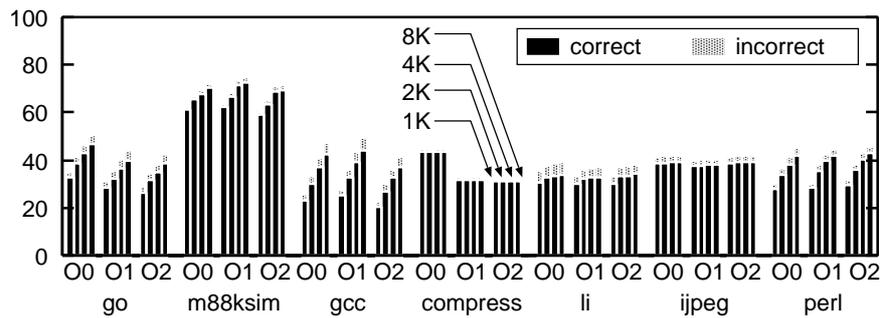


図 3 %予測可能性 (MIRV/最終値)

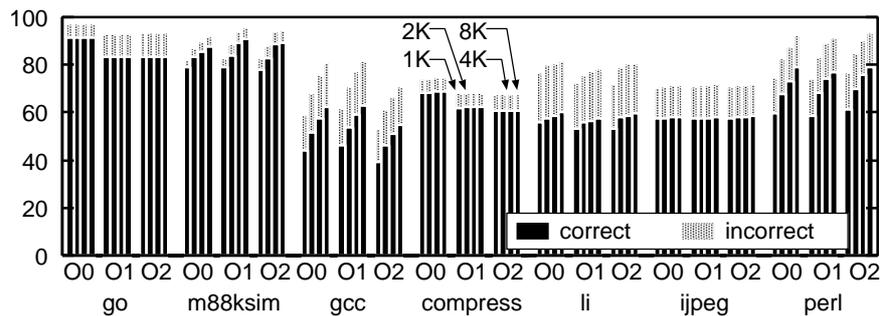


図 4 %予測可能性 (MIRV/ハイブリッド)

図 3 は MIRV コンパイラが生成したバイナリに最終値型予測器を試したときの結果である。GCC バイナリの場合と異なった結果も観測される。第 1 に、GCC バイナリの場合と比較して、一般に予測可能性と予測率が大きい。このことは全てのプログラムと全ての最適化レベルで観察される。表 1 と表 2 で見たように、最適化レベルが-01 または-02 の場合には、二つのコンパイラで生成されたバイナリの実行命令数には大差が無い。したがって、MIRV バイナリの方がデータ投機実行の恩恵を大きく得られると想像できる。第 2 に、最適化レベルが上がると、一般に若干予測可能性と予測率が小さくなる。GCC バイナリとの違いが顕著なのは 099.go である。しかし、この予測可能性と予測率の低下は GCC バイナリの 129.compress の場合と比較すれば、取るに足りない程小さい。したがって MIRV バイナリについても、最終値型予測器は最適化レベルが上がっても有効であると言える。

図 4 はハイブリッド予測器を用いた結果である。099.go の場合を除いて、最終値型の時と同様の傾向が見られる。絶対値は大きいものの、最適化レベルがあがると、若干予測可能性と予測率が低下する。ただし、099.go の場合には-00 から-01 に上がった時に著しい低下となっている。しかし、この予測可能性と予測率の低下は GCC バイナリの 129.compress の場合と比較すれば、やはり取るに足りない程小さい。したがってハイブリッド予測器も最適化レベルが上がっても有効であると言える。

4.2 プロセッサ性能

続いてプロセッサ性能への影響を調べる。プロセッサ性能の評価には、実行サイクル数を用いる。したがって、

値が小さい程性能が高いことになる。すべての場合で、最適化レベル-00 のバイナリをデータ値予測無しで実行したときのサイクル数で正規化した結果を示す。予測表の容量は 4096 エントリのみを評価した。

図 5 は GCC バイナリに最終値型予測器を適用したときの結果である。横軸はプログラム名と最適化レベルを、縦軸は相対的なプロセッサ性能を示している。2本のグラフからなる各グループは、左(図中灰色)がデータ値予測を行わない場合で、右(図中黒色)が予測を行った場合の結果である。まずデータ値を予測しない場合を観察すると、容易に判るように、最適化レベルが-00 から-01 になると著しく性能が向上している。続いてデータ値予測を行う場合を観察する。まず、最適化レベル-00 のバイナリにデータ値予測を行っても、最適化レベル-01 のバイナリで予測無しの場合に遠く及ばない。このことは、古くから存在するようなそれほど最適化の施されていないバイナリに対しては、データ値予測を行っても得られる恩恵が小さいことを意味している。つまり、ハードウェアに改良を加えるよりも、ソースプログラムを再コンパイルの方が効果的であると言える。続いて、最適化レベル-01 のバイナリで予測を行う場合と最適化レベル-02 のバイナリで行わない場合を比較すると、多くの場合で前者のほうが性能が高い。このことは、コンパイラが十分最適化を施したとしても、データ投機実行による性能向上の余地が十分あることを示している。第 3 にデータ投機実行の効果だけを見ると、最適化レベルが小さい程、プロセッサ性能の向上が大きいことが判る。これは、最適化の施されていないバイナリの方がデータ値を正しく予測できる命令数が多いという結果と一致する。

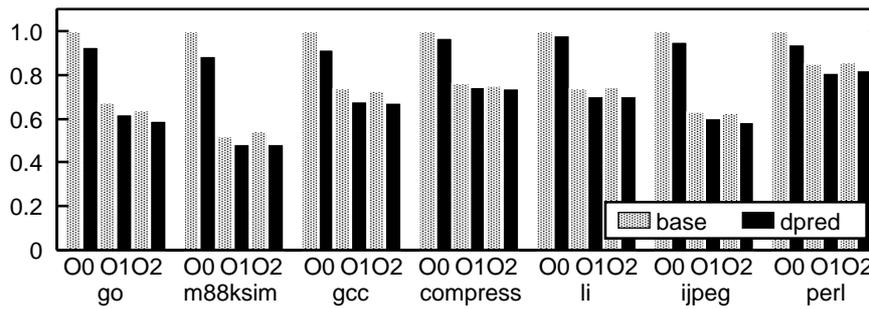


図5 プロセッサ性能 (GCC/最終値)

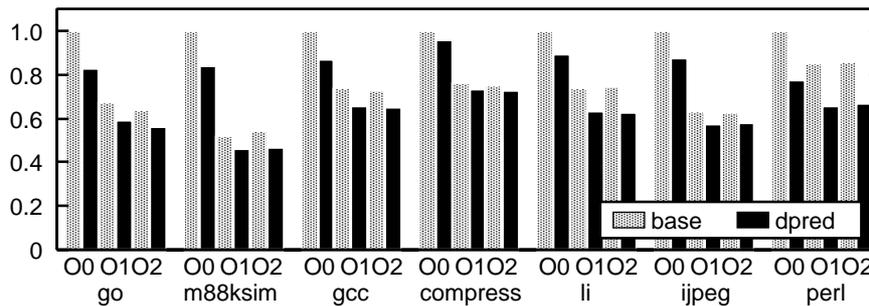


図6 プロセッサ性能 (GCC/ハイブリッド)

図6はハイブリッド予測器の場合である。全体として、データ値予測の貢献度は大きくなるが、一般的な傾向は最終値型の場合と似通っている。ただし 134.perl の場合が特別で、第1に最適化レベル-00のバイナリにデータ値予測を行うと、最適化レベル-01のバイナリで予測無しの場合よりも性能が優れている。第2に、最適化レベルが上がっても、性能向上の貢献度はほぼ一定である。

図7と図8は、MIRVバイナリにそれぞれ最終値型予測器とハイブリッド予測器を適用した時の結果である。まずデータ値を予測しない場合を観察すると、容易に判るようにGCCバイナリの場合とは異なり、最適化レベルが-00から-01に変わっても性能には大きな差は見られない。これはすでに表2で見たように、MIRVでは最適化レベル-00でも比較的積極的な最適化が施されているためである。両者の差が小さくなっているとしても、それでもなお、最適化レベル-00のバイナリにデータ値予測を行っても、最適化レベル-01のバイナリで予測無しの場合には及ばない。このことから、最適化のレベルが低い古くから存在するバイナリに対しては、ハードウェアによるデータ投機実行に頼るよりも、ソースプログラムを再コンパイルする方が効果的であると確認できる。この点を除くと、絶対値は異なるものの傾向としてはGCCの場合と同様の結果が確認できる。したがって今回調査した範囲では、使用されるコンパイラや最適化レベルに関係なくデータ値予測は有効であると言える。

5. まとめ

本稿ではGCCとMIRVコンパイラを用いて最適化レベルを変化させた時の、生成されるバイナリの予測可能

性への影響を調査した。近年、より多くのILPを抽出するためにデータ依存を投機的に解消する方式が注目されている。ある命令の結果は値予測器を用いて予測され、この命令と依存関係にある命令を同時に実行できる。したがって、ILPを積極的に抽出できるわけである。しかし現在のところ、データ値予測器に関する検討は主に予測方式に関してである。われわれの知る限りにおいては、コンパイラによる最適化とデータ値予測可能性との間の関係を調査した報告は無い。もしコンパイラが十分最適化を施し、データ値予測に基づいたデータ投機実行による性能向上の余地が無いとすれば、データ投機実行は無意味である。そこで本稿では、コンパイラの最適化レベルを変化させてコンパイルされたバイナリを用いて、これらの間の関係を調査した。詳細なシミュレーションを実施した結果、どちらのコンパイラを用いても、最適化レベルが上がっても予測可能性は十分あることが確認できた。また、プロセッサへの影響も調査した。こちらも同様の結果が得られ、コンパイラが十分最適化を施しているとしても、データ値予測に基づいたデータ投機実行による性能向上の余地は十分にあることが判った。

本報告に関する将来の課題の一つは、他のコンパイラ、特に定評のある商用のコンパイラであるDEC Cコンパイラを用いた調査を行うことである。-Oオプションを用いるだけではなく、細かく最適化を使い分けたと時の影響を調べることも興味深い。また、最適化レベルが低いバイナリに対するデータ投機の効果は再コンパイルに及ばないことを鑑みると、バイナリの実行時動的最適化とデータ投機実行の組合せ¹⁴⁾を検討することも必要である。

謝辞

本研究の一部は、科学研究費補助金 奨励研究(A) 課題

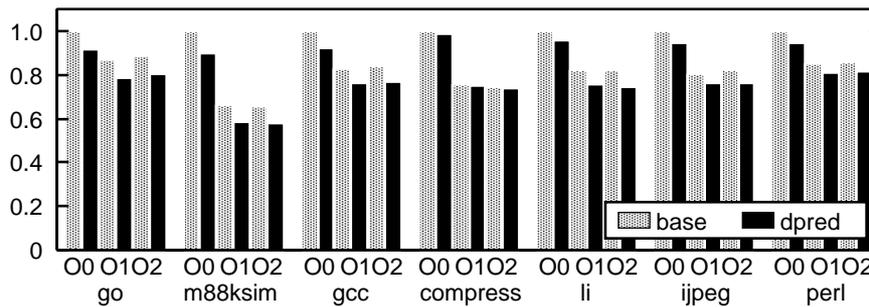


図7 プロセッサ性能 (MIRV/最終値)

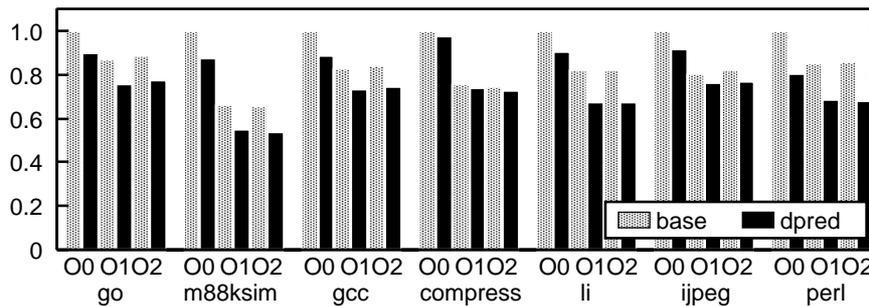


図8 プロセッサ性能 (MIRV/ハイブリッド)

番号 12780273, 福岡県産業・科学技術振興財団 テーマ探索・シーズ発掘事業の援助によるものです.

参考文献

- Burger D., Austin T.M.: The SimpleScalar tool set, version 2.0. ACM SIGARCH Computer Architecture News, 25(3) (1997)
- Calder B., Reinman G., Tullsen D.M.: Selective value prediction. 26th Int'l Sym. on Computer Architecture (1999)
- Del Pino S., Pinuel L., Moreno R.A., Tirado F.: Value prediction as a cost-effective solution to improve embedded processor performance. 3rd International Meeting on Vector and Parallel Processing (2000)
- Fu C-y., Jennings M.D., Larin S.Y., Conte T.M.: Software-only value speculation scheduling. Technical Report, Dept. of Electrical and Computer Engineering, North Carolina State University (1998)
- Gabbay F.: Speculative execution based on value prediction. Technical Report #1080, Dept. of Electrical Engineering, Technion (1996)
- Lipasti M.H., Shen J.P.: Exceeding the dataflow limit via value prediction. 29th Int'l Symp. on Microarchitecture (1996)
- Morancho E., Llaveria J.M., Olive A.: Split last-address predictor. Int'l Conf. on Parallel Architectures and Compilation Techniques (1998)
- Postiff M., Greene D., Lefurgy C., Helder D., Mudge T.: The MIRV SimpleScalar/PISA compiler. Technical Report CSE-TR-421-00, Dept. of Computer Science, University of Michigan (2000)
- Rychlik B., Faistl J.W., Krug B.P., Kurland A.Y., Sung J.J., Velez M.N., Shen J.P.: Efficient and accurate value prediction using dynamic classification. Technical Report CMuART-98-01, Dept. of Electrical Computer Engineering, Carnegie Mellon University (1998)
- Sazeides Y., Smith J.E.: Implementations of context based value predictors. Technical Report TR-ECE-97-8, Dept. of Electrical Computer Engineering, University of Wisconsin-Madison (1997)
- Sohi G.S.: Instruction issue logic for high-performance, interruptible, multiple functional unit, pipelined computers. IEEE Trans. Comput., 39(3) (1990)
- Tullsen D.M., Seng J.S.: Strageless value prediction using prior register values. 26th Int'l Symp. on Computer Architecture (1999)
- Wang K., Franklin M.: Highly accurate data value prediction using hybrid predictors. 30th Int'l Symp. on Microarchitecture (1997)
- 佐藤寿倫, 有田五次郎: KIT COSMOS プロセッサ: 背景と着想. 信学技報 CPSY99-115 (2000)
- 杉谷樹一, 濱野彰彦, 佐藤寿倫, 有田五次郎: 値予測におけるコンパイラが与える影響の違い. 電気関係学会九州支部 53 回連合大会 (2000)
- 濱野彰彦, 杉谷樹一, 佐藤寿倫, 有田五次郎: プログラムの最適化による値予測実行率への影響. 電気関係学会九州支部 53 回連合大会 (2000)