逐次型推論マシンＣＨＩの性能評価

5B-9

梅村　護、中崎良成、小長谷明彦、幅田伸一、新　　淳
日本電気㈱　Ｃ＆Ｃシステム　研究所

1．はじめに

筆者等は第５世代ﾌﾟﾛｼﾞｪｸﾄの一環としてICOTより委託を受け、述語論理型言語を基本とする逐次型推論ﾏｼﾝ CHI の開発を担当した。[1] CHIのｱｰｷﾃｸﾁｬ の特徴は、平均遅延時間0.7ns のCML を用いたﾏｲｸﾛｻｲｸﾙ100ns の ﾏｼﾝであり、しかも大容量の記憶システムを備えて述語論理型言語の引数のﾀｲﾌﾟ に依存して内部ﾚｼﾞｽﾀ群を効率よく利用した高速なﾕﾆﾌｨｹｰｼｮﾝ 処理[2] を実現することにある。 CHIはさらに述語論理型言語処理の特徴を考慮し,ﾀｸﾞｱｰｷﾃｸﾁｬを採用している。本稿ではその基本命令の実行時間、ﾍﾞﾝﾁﾏｰｸﾌﾟﾛｸﾞﾗﾑ の実行時間およびｴｷｽﾊﾟｰﾄｼｽﾃﾑを実行させたときの処理時間の実測値を他のﾏｼﾝ における実測値と比較して報告する。

2．基本命令の実行時間

表１は述語論理型言語をﾚｼﾞｽﾀ方式で実現するための基本命令実行時間をﾏｲｸﾛｻｲｸﾙ単位で示している。ここで get命令はﾚｼﾞｽﾀに格納されている引数の値によりﾕﾆﾌｨｹｰｼｮﾝ を行う命令、 put命令は引数値を対応するﾚｼﾞｽﾀに格納する命令、 unify命令は構造体ﾃﾞｰﾀ 中の対応部分についてのﾕﾆﾌｨｹｰｼｮﾝ を行う命令であり、いずれも述語論理型言語処理におけるﾕﾆﾌｨｹｰｼｮﾝ 処理を実現するための基本となる命令である。

実測は、ﾊｰﾄﾞｳｪｱﾚｼﾞｽﾀの個数(32)だけ命令を実行させ、その間のｸﾛｯｸを計数して32で割る方法で実施した。実測に際しては命令およびﾃﾞｰﾀ は全てｷｬｯｼｭﾒﾓﾘに格納した後に計測を開始する。

表１．ＣＨＩ基本命令実行時間実測値（単位100ns）

Instruction	Time	R	W
get-constant (var-cst)	12	●	
get-constant (cst-cst)	6	●	
get-nil (var-nil)	9		●
get-nil (nil-nil)	4		
get-list (var-list)	9.2		●
get-list (list-list)	2		
get_permanent_variable	3.5		●
get_permanent_value (var-int)	9		●
get_permanent_value (int-int)	7	●	
get_temporary_variable	2		
get_temporary_value (var-int)	9		●
get_temporary_value (int-int)	6		
put-constant	4	●	
put-nil	2		
put-list	3		
put_permanent_variable	9.0		●
put_permanent_value	4	●	
put_temporary_variable	9.0		●
put_temporary_value (var)	6		
put_temporary_value (int)	3		
put_unsafe_value	4	●	
put_unsafe_value (var)	8.4	●	
put_unsafe_value (var: dealloated)	12.6		●
unify-constant (var-cst)	11	●	●
unify-constant (cst-cst)	8	●	
unify-nil (var-nil)	9	●	●
unify-nil (nil-nil)	5	●	
unify_permanent_variable	5	●	●
unify_permanent_value (var-int)	9.0	●	●
unify_permanent_value (int-int)	6.8	●	
unify_temporary_variable	4	●	
unify_temporary_value (var-int)	11.4	●	●
unify_temporary_value (int-int)	6	●	
unify_void	2		

表中、Ｒ欄は ﾒﾓﾘ読み出しをW欄は ﾒﾓﾘ書き込みを伴う事を示す。

Performance Evaluation of the Sequential Inference Machine : CHI

Mamoru UMEMURA,Ryosei NAKAZAKI,Akihiko KONAGAYA,Shin-ichi HABATA,Atsushi ATARASHI
NEC Corporation

CHIのｷｬｯｼｭﾒﾓﾘはWRITE THROUGH 方式であり、主記憶はｲﾝﾀｰﾘｰﾌﾞ 方式で実現されている。したがってﾒﾓﾘ のｱｸｾｽを伴う場合にはｱﾄﾞﾚｽとｱｸｾｽのﾀｲﾐﾝｸﾞ に依存して実行時間にﾊﾞﾗつきが生じる。表中の端数はこの現象を示している。

表１に示したように各命令はそれぞれﾒﾓﾘ･ｱｸｾｽの有無とｱｸｾｽ方向（READ/WRITE）がわかっている。ﾒﾓﾘｱｸｾｽ を伴わないときの実行時間は２～４ｻｲｸﾙ、読みだしを伴うときには４～６ｻｲｸﾙ、書き込みを伴うときには９～12ｻｲｸﾙを要している。書き込みを伴うときの実行時間のﾊﾞﾗつきは、ｲﾝﾀｰﾘｰﾌﾞ が理想的に効果を発するときとそうでないときを反映している。

３．ベンチマークプログラム実行性能

表２は第１回Prologｺﾝﾃｽﾄ [3] で用いられたﾍﾞﾝﾁﾏｰｸﾌﾟﾛｸﾞﾗﾑ の中、代表的なものの実行時間を示す。CHI の実行時間は他の処理系に比べ、２～５倍以上の高い性能を示している。

表２．ベンチマークプログラム実行時間（単位ms）

プログラム	ＣＨＩ	DEC2060	SYMBOLICS
Reverse30	2.49	13.8	9.4
Q-Sort 50	6.64	17.1	12.3
8-Queens	31.39	128	65.3
8-Queens-All	506.0	2019	1059

４．応用プログラム

表３は CHI上で計算機構成支援ｴｷｽﾊﾟｰﾄｼｽﾃﾑ[4] を実行したときの処理時間を示す。この応用ﾌﾟﾛｸﾞﾗﾑ はACOS1000のShapeUp[5]という拡張Prolog処理系を用いて実現されたｼｽﾃﾑを CHIのﾌﾟﾛｸﾞﾗﾑ に書き直したものである。その処理内容は、計算機を構成する要素（CPU,Disk等）の接続方法に関してﾏﾆｭｱﾙ には表現されていない専門家のﾉｳﾊｳを ﾙｰﾙとして記述し、それを結び合わせることによって正しい接続を出力するｼｽﾃﾑである。各処理ﾌｪｰｽﾞのうち、ﾃﾞｰﾀ 入力およびﾃﾞｰﾀ 出力処理は、ﾌｧｲﾙへのｱｸｾｽ部分であり、ほとんど改善は見られないが、中心部である ﾙｰﾙの探索と接続候補の検索処理は約３倍の高速化が実現されている。ShapeUp は言語Ｃで書かれたｾﾐ･ｺﾝﾊﾟｲﾙ方式の処理系であり、約15MIPSのACOS1000上では高速な処理性能を発揮する。

表３．応用プログラム処理時間比較（単位秒）

処　理	ＣＨＩ	ACOS1000	相対倍率
データ入力	0.81	0.89	1.10
階層構成	2.18	6.82	3.13
製品再構成	2.00	5.85	2.93
データ出力	3.56	3.46	0.97

表３の実測に於いてはShapeUp ﾌﾟﾛｸﾞﾗﾑ をほぼそのままの形で移植したが、 CHIの特徴を生かして最適化を行えば更に処理性能を改善することができる。

５．むすび

CHI の性能評価結果について述べた。筆者等は引き続き第５世代ﾌﾟﾛｼﾞｪｸﾄの一環で CHIの小型化版を実現中であるが、本稿で述べた評価結果を踏まえ、より高度な述語論理型言語処理環境をﾕｰｻﾞ 各位に提供できるよう研究を進めている。最後に本研究の機会を与えて頂き、全面的な御支援を頂いているICOTの内田室長はじめ関係各位に深謝する。

参考文献

1]山本他；「高性能Prologﾏｼﾝ:CHI とそのﾌﾟﾛｸﾞﾗﾐﾝｸﾞ環境」情報処理学会ｺﾝﾋﾟｭｰﾀｼｽﾃﾑｼﾝﾎﾟｼﾞｳﾑ 60年12月

2] Warren,D.H.; An Abstract Prolog Instruction Set , TR309, Artificial Intelligence Center, SRI International, 1983

3]奥乃；「第３回Lispｺﾝﾃｽﾄ および第１回Prologｺﾝﾃｽﾄ報告」情処記号処理研究会資料85-SYM-33 1985

4]出口他;「構成支援ｴｷｽﾊﾟｰﾄｼｽﾃﾑに於ける知識表現」情報処理学会第30回全国大会5L-8, 1985

5]梅村他;「文字列処理を導入したProlog : ShapeUpについて」The Logic Programming Conference'83