A-NET マルチコンピュータにおけるメッセージ処理の高速化

2G－4　澤田東　吉永努　馬場敬信
宇都宮大学工学部*

1. はじめに

A-NET マルチコンピュータは、並列オブジェクト指向言語 A-NETL からトップダウンに設計した分散メモリ型並列計算機である [1]。現在、16 ノードプロトタイプマシンが動作中であり、ハードウェア、ソフトウェアの通信性能の評価を行なってきた。この結果、ハードウェアレベルでの通信性能は十分高速であるが、言語レベルから見るとメッセージの処理が重いことが分かった [2]。

このため、我々は、A-NET マルチコンピュータがプロトタイプマシンであることを活用して、A-NET ローカル OS で行なっていたメッセージ処理をファームウェア化することにより、処理の高速化を図った [3]。

2. メッセージ処理高速化の方針

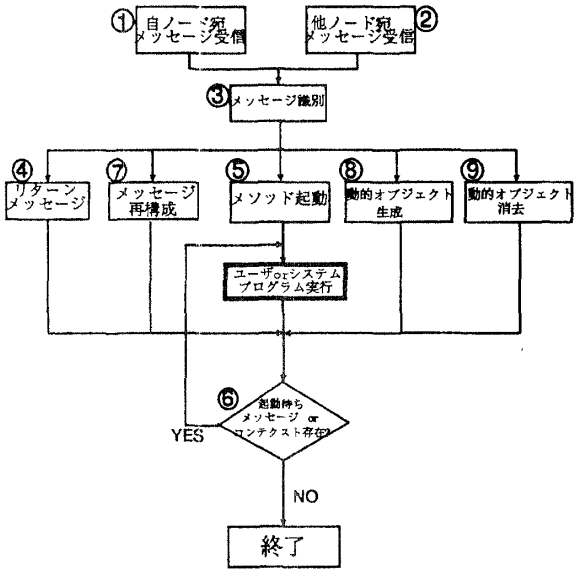

図 1: OS のメッセージ受信処理

A-NET ローカル OS の行なうメッセージ処理は、図 1のようになる。これらを効果的にファームウェア化するために、各処理の実行回数、実行時間を、8Qs と台形公式のよる定積分 (Q11) の 2 つのアプリケーションプログラムを用いて調べた。

表 1、表 2に図 1の①~⑥までの処理の実行回数を示す。これはアプリケーションプログラム開始から終了までの値である。また、表 3に、8Qs の 1 つのメッセージと Q11 はループに対応する実行時間を示す。

8Qs は、8 個の配列と row を表す変数の 2 つを引数とする、メッセージ分割のない比較的小さな過去型メッセージを使用する。システムの割合は50%強である。これらの表からメッセージによるメソッドの起動が実行回数、実行時間がともに大きく、ここを高速化すると効率的であることが分かる。

表 1: 8Qs の各処理の実行回数 (1node あたり)

処理	①	②	③	④	⑤	⑥
2nodes(回)	245	249	494	0	493	1
8nodes(回)	15	110	125	0	124	1

表 2: Q11 の各処理の実行回数 (1node あたり)

	①	②	③	④	⑤	⑥
2nodes(回)	500	6	506	250	254	251
8nodes(回)	125	4	129	63	67	63

表 3: 8Qs、Q11 における処理毎の実行時間

	①	②	③	④	⑤	⑥	user
8Qs(μs)	0	148	75	0	508	43	744
Q11(μs)	145	0	154	166	223	29	101

Q11 は、メインオブジェクトから各インデックストオブジェクトに初期値が渡されてから、各インデックストオブジェクトがそのノード内で自ノード宛のメッセージを未来型で送信する。引数は構造体を持たない小さなものであり、またそれにより起動されるメソッドの粒度も小さいため、システムの割合は 80%強と非常に大きくなっている。この問題も、メッセージによるメソッドの起動が重く、また、リターンメッセージの処理も大きいことが分かる。表ではメッセージ受信も重く見えるが、これは 1 ループ中で複数この処理を行なうためである。

また、8Qs や Q11 では使用しなかった⑦~⑨の OS 処理についても実行時間を調べた。メッセージ再構成にかかる実行時間を調べるために、90 個の配列要素を持つ大きなメッセージを、他のオブジェクトに送信した。A-NETL の 1 語は 5 バイトであり、ルータの最大パケット長は 255 バイトであるため、このメッセージは、メッセージ送信機械命令により、3 つのパケットに

*Speedup of Message Handling on the A-NET Multicomputer,Akira SAWADA, Tsutomu YOSHINAGA,and Takanobu BABA,Utsunomiya University.

分割される。これらの実行時間を表4に示す。これより、メッセージ再構成は、表の処理をパケットに分割された数だけ繰り返すことになり、非常に重い処理であることが分かる。

表 4: 処理⑦~⑨の実行時間

処理	⑦	⑧	⑨
(μs)	1503.8	561.0	342.2

3. 高速化の効果

まず、図1の処理を個別に高速化した時の効果を述べる。ここでは、OS と同等の処理を機械命令 trap によりマイクロプログラムに移行し、実行した。

表5に 8Qs と Q11 について部分的に OS を高速化した時の効果を示す。また、表6に 8Qs と Q11 では使用しない⑦~⑨の OS 処理についての効果を示す。この結果から前節の予備評価結果から予測されたように、メッセージによるメソッド起動をファームウェア化した時の効果が最も大きいのが分かる。

8Qs が、Q11 より大きな高速化の効果が得られたのは、表3で②と⑤ の割合の大きい 2 つの処理をファームウェア化できたことによる。

表 5: OS 部分的ファームウェア化の効果 (その 1)

		高速化前	②	④	⑤	ALL
8Qs	ms	93.5	90.1	93.5	65.8	62.1
	比	1	1.04	1	1.42	1.51
Q11	ms	92.42	92.27	87.72	84.66	79.81
	比	1	1	1.05	1.09	1.16

表 6: OS 部分的ファームウェア化の効果 (その 2)

	高速化前 (ms)	高速化後 (ms)	比 (倍)
⑦	1.50	0.33	4.50
⑧	0.56	0.29	1.95
⑨	0.34	0.23	1.50

次に、図1の全処理をファームウェア化した時の効果を表7に示す。ここでの結果は、部分的ファームウェア化の効果と比較し、8Qs と Q11 で逆転が起こっている。これは、もともとのユーザの割合が 8Qs の方が大きいことによる。また、メッセージ再構成、オブジェクト生成 、オブジェクト消去では 13~18 倍と非常に大きな効果が得られた。これは、これらの処理にはユーザが含まれず、すべての処理についてファームウェア化ができたことによる。

4. 考察

ファームウェア化による高速化の要因は、次の 3 点である。

表 7: 全 OS ファームウェア化の効果

		高速化前	高速化後	比 (倍)
8Qs	(ms)	93.49	49.47	1.89
Q11		92.42	17.81	5.19
⑦	(μs)	1504	88.33	17.02
⑧		560	40.50	13.85
⑨		340	18.83	18.17

第 1 に、OS は高機能な言語 A-NETL で記述されているが、OS では、動的な型の検査などをする必要はなく、ファームウェア化の際は、静的に決定できる情報を最大限に活用して負荷を抑えた。

第 2 に、プロトタイプマシンのマイクロアーキテクチャは、ALU 演算やメモリアクセス、タグ操作などを同時に制御できる VLIW 的な要素を持っているが、A-NETL レベルでは、これを生かすことができない。これにより、機械命令で 2 もしくは 3 命令で 50~100 マシンサイクル程かかっていた処理をファームウェア化により 1 マシンサイクルに抑えられる場合があり、これを活用して負荷を抑えた。

第 3 に、A-NET マルチコンピュータは、オンチップメモリを前提に、高速なメモリ-メモリ演算を想定しており、プロトタイプマシンでは、オペランドのフェッチ、アドレス演算、データのメモリへの読み書きなどが負荷となっている。OS の高速化では、これらの負荷が最小限に軽減できたことも高速化の要因となっている。

6. おわりに

本研究では、A-NET ローカル OS によるメッセージ処理をファームウェア化することにより、アプリケーションプログラムで約 2~5 倍の速度向上が得られた。

今後の課題として、今回は触れなかったローカル OS の実行順序制御の機構についてもファームウェア化を行ない、さらなる高速化を目指す予定である。

謝辞

本研究は、一部文部省科学研究費、基盤 (C)08680346 および電気通信普及財団の援助による。

参考文献

[1] 馬場, 吉永: "並列オブジェクト指向トータルアーキテクチャA-NET における言語とアーキテクチャの統合", 信学会論文誌, Vol.J75-D-I, No.8, pp.563-574(1992).

[2] 廣田守, 吉永努, 馬場敬信: "A-NET マルチコンピュータにおける適応型ルータの通信性能" 情処学計算機アーキテクチャ研報 96-ARC-119, vol.96, no.80, pp.31–36 (1996).

[3] 岩本善行, 吉永努, 馬場敬信: "言語レベルからみた A-NET マルチコンピュータのメッセージパッシング性能" 情処学計算機アーキテクチャ研報 96-ARC-119, vol.96, no.80,