オペレーティング・システム 48－5
（1990．9．7）

A-NETローカルOSのメッセージ受理機構

吉永 努　　馬場 敬信

宇都宮大学 工学部 情報工学科

A-NET ローカルOS第2版の設計方針とシミュレータを用いた実験結果について報告する．第2版では，特にメッセージの受理処理を高速化した．第1版との比較から，引数なしのメッセージを受理してからユーザのメソッドを起動するまでの時間は約40%高速化された．また，引数を1つ伴うときのOSの実行時間は，約17～49%短縮された．さらに，サンプルプログラムを用いたシミュレーション結果から，(1) ユーザとシステムの実行時間の比は1:9.7である，(2) アドレス化セレクタによって平均2.9回の辞書探索が不要となった，(3) 最大並列度は，ほぼ使用PE数に近い値が得られる，などが分かった．

Message Reception Mechanism of the A-NET Local Operating System

Tsutomu Yoshinaga and Takanobu Baba

Department of Information Science, Faculty of Engineering

Utsunomiya University, Utsunomiya 321, Japan

This paper describes the A-NET local operating system version 2 and the experimental results. The intention of version 2 is to achieve fast processing of the message reception. When the OS receives a message with no argument, it performs the operation with about a 40% reduction in the processing time compared with OS version 1. For a message with one argument, there is about a 17 to 49% reduction in execution time. Experimental results show that: (1)the execution time ratio between user and OS is about 1:9.7, (2)the translation of a selector to the corresponding address saves 2.9 times the message dictionary search, and (3)the maximum parallelism is close to the number of used PEs.

1．はじめに

我々は，並列オブジェクト指向実行モデルを採用したプログラミング言語 A-NETL を設計し，現在，その実行を支援するための A-NET 高並列計算機の開発を進めている[1]．A-NET 計算機の各ノードプロセッサは，260KB のローカルメモリを持つメソッド実行用の40ビットPEと，メッセージ通信用のルータ，60KB のPE-ルータ間共有メモリからなる．PE-ルータ間にメッセージ入出力キューを設けて内部処理と通信を分離することにより，スループットの向上を図っている[2,3]．全体のノード数は数百～数千を目標とし，ルータの構成をネットワークトポロジ独立なものとするとともに，オブジェクト間の関係に基づき各オブジェクトを物理的ネットワークに最適に割り付けるアロケータを開発して，各種応用問題に適応できるようにしている[4,7]．

ローカルOSは，各PE上でオブジェクトの記憶管理，メッセージの受理，A-NETL のプログラミングパラダイムを反映したコンテクスト管理等を行う[5]．今回，メッセージを受理してからユーザのメソッドを起動する処理の高速化などを目的とした改良を行った．本稿では，このローカルOS第2版の改善方針と実験結果を中心に述べる．

2．ローカルOS第2版の改善方針

A-NET ローカルOSの第一義的な目的は，A-NETL の持つ並列オブジェクト指向実行モデルの特徴を損なうことなく，その実行のオーバヘッドを極力抑えることにある．このオーバヘッドには2種類のものが考えられる．すなわち，オブジェクト指向であるためのものと，実行の並列化によるものとである．前者には，実行順序の非決定性やメッセージ駆動方式によるものなどがあり，後者には，同期構文の処理や通信遅延などがある．これらの与える影響は，応用プログラムによって様々であるが，言語処理系の工夫やハードウェアサポートとともに，OSの機構によってできる限り多くのプログラムについて実行性能を上げたい．我々は，第1版OSを用いたいくつかのサンプルプログラムのシミュレーション結果を基に，OSとユーザとの処理のバランスなどを検討し，特にメッセージを受理してからユーザのメソッドを起動するまでの高速化とコンテクストチェンジの高速化が必要であると考え，ハードウェアコストとのトレードオフを考慮して，命令セットとハードウェア構成に若干の変更を加え，第2版OSを設計した．また，並列プログラムのデバッグ機能の強化も図っている．

第2版の主な改善点を以下に示す．

（1）メソッド起動の高速化

PEに割り付けられたオブジェクトは，メッセージによって活性化される．ルータは，自ノード宛メッセージを受け取ると，これをPE-ルータ間共有メモリ上の入力メッセージキューに書き込み，PEに割り込みをかける（メッセージ割り込み）．PEは，メッセージ入力待ちのアイドル状態，もしくはユーザメソッド実行状態においてこの割り込みを受け付け，OSのメッセージ受理メソッドを起動して，システム状態に移行する．この関係を図1に示す．ここで，システムがユーザを起動するまでの時間と起動されるユーザ処理の粒度のバランスが問題となるが，並列化による処理の高速化を期待する場合には，できるだけシステム時間を短くしたい．このため，新たにメッセージレシーブ命令を定義し，ローカルメモリ上にコンテクストを生成せずに，入力キュー内のメッセージを直接実行できるようにする．

また，静的に定義されたオブジェクトに送信するメッセージのセレクタをアドレス化し，メッセージ辞書の探索をせずに起動するメソッドを決定できるようにする．

（2）コンテクストチェンジの高速化

入力キューに格納されたメッセージは，基本的に先頭から（到着順に）実行されるが，送信済みメッセージの取り消しや，オブジェクトの動的消去などをサポートするために，ユーザ状態ではメッセージ割り込みを受け付け，システム状態に移行する．このとき，ユーザの実行イメージを待避しなくてもすむよう，ユーザ用とシステム用の2つの特殊レジスタセットを設ける．図2にレジスタセット，コンテクスト，メッセージの構造を示す．これらは，待避と復帰をブロック転送できる

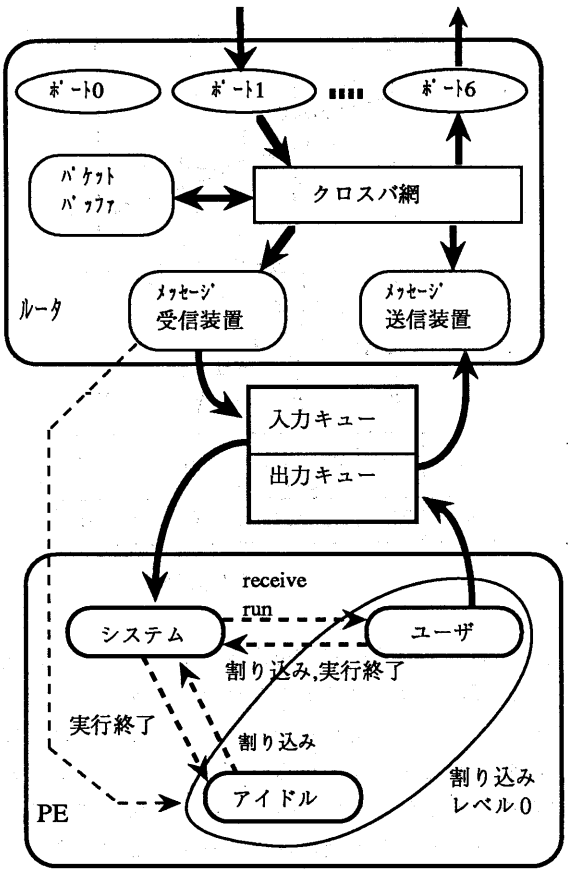


図1．メッセージ送受信とＰＥの状態遷移

よう，対応した構造を持つ．割り込みによって中断されたユーザプログラムの実行は，実行イメージがレジスタ上にある場合にはＰＣとフラグレジスタの設定だけで再開することができる．また，同期のためにコンテクストに待避される実行イメージの大きさは，メッセージを実行しているときは９語，１度待避されたコンテクストを実行しているときは５語である．

（３）メッセージ引数再構成の高速化

A-ＮＥＴＬ では，メッセージによる引数渡しは，構造体データを含めてすべてコピー渡しである．そこで，第１版では図３(a)に示すように，メッセージのサイズを抑えるため，リストや配列などのデータを引数とする場合，これをメッセージ中に圧縮して送信し，受信側のＯＳでローカル

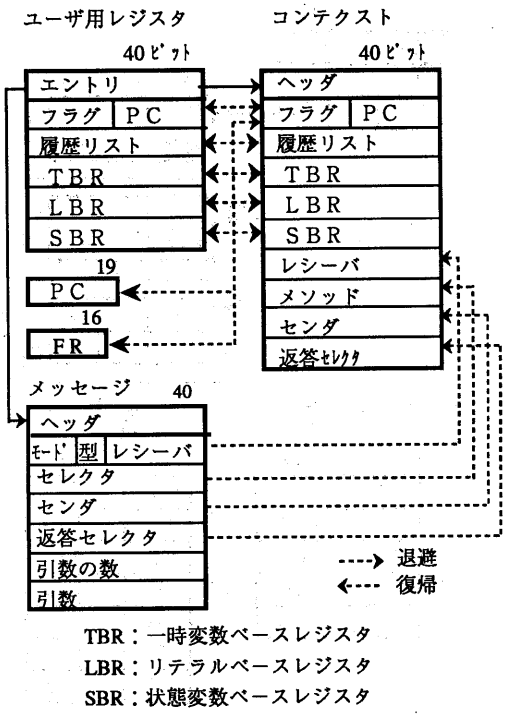
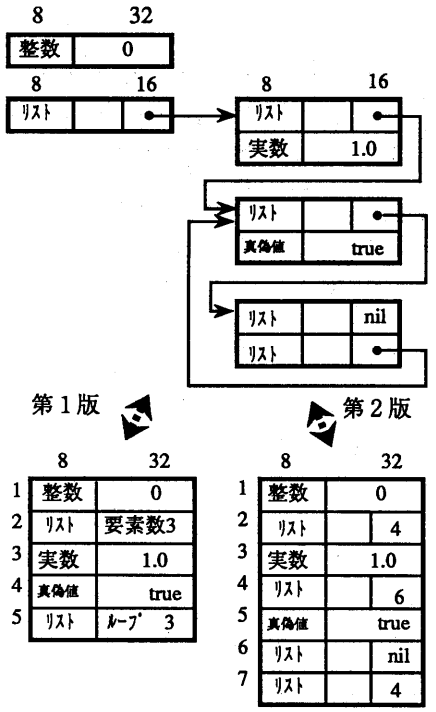


図２．レジスタへの退避と復帰

メモリに展開していた．すなわち，世代別ガーベジコレクションのためのフラグや次のセルへのポインタを含む構造体データセルのヘッダを省略したものから，受け側で引数のデータ型を調べ，各データセルごとにコピーして元のデータ構造を再現していた．送信側ノード上でのデータの世代もポインタも受信側ノード上では不要となるからである．しかし，第１版を用いたシミュレーション結果から，数語の転送時間とバッファ領域を節約するよりも，その再構成にかかる時間を短縮することが重要であると判明したため，第２版では図３(b)に示すように，構造体セルをヘッダ付きで送信するようにする．メッセージ送信時，ヘッダ部に含まれるポインタは，ベースレジスタ TBR のオフセットに変換される．これにより，受信時には引数の構造を詳しく識別することなく，入力キューからローカルメモリに一括コピーして再配置できる．

引数再構成の手数を減らすことにより，(1)の



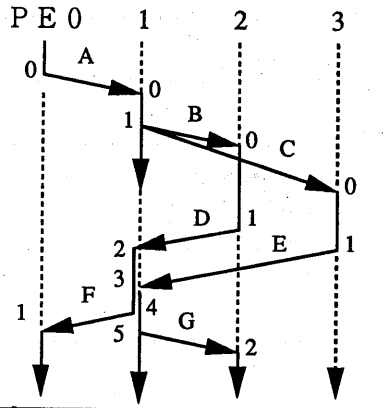
(a) 圧縮した引数　(b) 圧縮しない引数

図3．メッセージの引数

メソッド起動とリターンメッセージの返値格納を高速化することができる．

（4）並列プログラムのデバッグ支援

A-NETL のデバッガは，ホスト上でユーザとのやり取りをするマスタデバッガと，各PE上のOSレベル／ファームウェアレベルのローカルデバッガからなる[6]．このうち，OSレベルのローカルデバッガは，ホストからのメッセージによる通常実行モードとデバッグモードの切り替え，ブレークポイントの条件設定などと，タイムスタンプ付きのメッセージ受信履歴の保存を行う．図4に，タイムスタンプ付きメッセージの送受信履歴の概念図を示す．タイムスタンプには，メッセージの送受信ごとに更新される局所的，論理的なものを使用する．送信履歴は，メッセージセンド命令を実現するファームウェアによって採られるが，送信時刻に加えて，そのメッセージ送信の原因となったメッセージの受信時刻も付す．これは，



送信	*A0	0B1,0C1, 3G5	0D1	0E1
受信	F1	A0,D2,E3	B0,G2	C0

図4．メッセージ送受信履歴

メッセージの因果関係を再現するためである．例えば図4では，PE1上のオブジェクトが，時刻0にメッセージAを受信し（A0），それによって起動されたメソッドから時刻1にメッセージBとCを送信した（0B1，0C1）ことを記録している．デバッグモードの設定は，ノードごとに可能であるが，複数のノードをデバッグモードで実行すれば，送受信履歴を照らし合わせることによって，全体のオブジェクト間でのメッセージ送受信関係を提示することができる．

3．メッセージの受理機構

OS第2版におけるメッセージ受理処理を詳細にみると，図5のようになる．

（1）PEがメッセージ割り込みを受け付けると，割り込みベクタに対応した番地からOSのメッセージ受理メソッドのアドレス化セレクタを読み出して，これを起動する．OSは，ルータによって共有メモリに書き込まれた入力メッセージのエントリ（先頭番地とサイズを含む）を読み出して，エントリを次の入力メッセージのものに書き換えるかクリアする．

（2）メッセージから宛先，型などを読み込んで，宛先オブジェクトがPE上に存在するか，メッセ

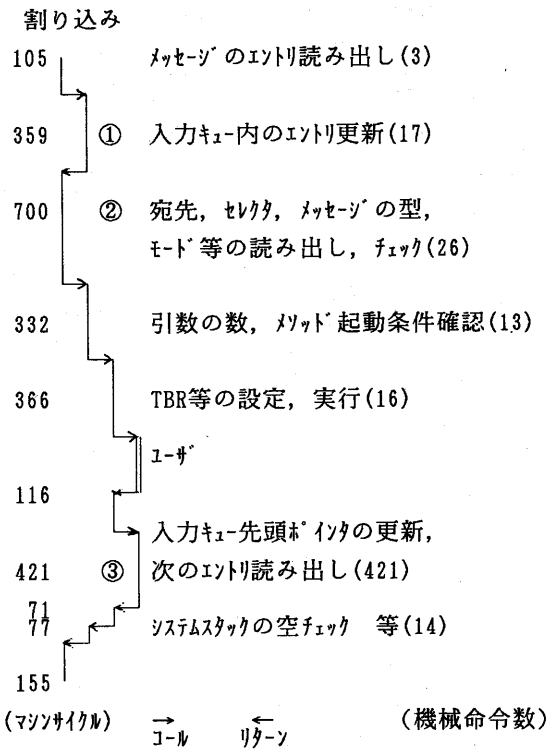


図５．メッセージ受理処理詳細

ージはメソッド起動のためのものかリターンかなどを調べる．このとき，オブジェクトはＩＤに含まれる番号から表検索されるため，１ノード上で複数のオブジェクトを管理している場合にも検索時間が増えることはない．また，セレクタがアドレス化されたものであれば，メッセージ辞書の検索を行わずにメソッドを見つけることができる．このとき，ＰＥがデバッグ実行モードであるか，メッセージがデバッグモードで送信されたものであれば，メッセージの内容をローカルメモリにコピーして受信履歴を作成する．

（３）仮引数と実引数の数を確認し，引数を一時変数領域にコピーする．

（４）ＰＥがデバッグ用ユーザ停止モードでなければ，メッセージレシーブ命令によりユーザメソッドを実行する．

（５）実行終了したメッセージが入力キューの先頭にあれば，これを削除してキューポインタを更新する．

この中で実行時間を消費しているところは，①，③の入力メッセージキュー操作と②のメッセージ内の各情報フィールドの切り出しとチェックである．メッセージキューには，デュアルポートメモリを仮定しているが，これは，ＰＥとルータの同時アクセスの他，メッセージを配列としてランダムアクセスするためである．ここでのオーバヘッドとしては，キューを循環バッファとして扱うことと，入力メッセージキューを２回見に行くことが挙げられる．キューを介したメッセージパッシングは，複数プロセッサ上での実行によるためのものであるが，これを単一プロセッサ上で，例えばスタックを用いて実行する場合でも，ほぼ同様の手数がかかると考えられ，実行モデルを簡単化せずに，これ以上劇的に手数を減らすことは難しいと思われる．

４．実験結果

４．１　メッセージ受理処理の評価

2.節で述べた改善案に基づきＯＳ第２版を実現し，シミュレータを用いてメッセージ受理処理の手数を第１版のものと比較した．使用したサンプルプログラムは図６に示すオブジェクトである．

表１に，主なシステム特権命令とその実行マシンサイクル数を示す．実行マシンサイクル数は，機械命令を実現するマイクロプログラムから求めたものである．各命令において，オペランドのアドレス計算，読み出しのために des×3，so×5 の時間がかかっている．追加した命令は，メッセ

```
object  test
state   s       " state variable"
methods {
    m0 {        " no arguments  "
        s = 0
    }
    m1: arg {   " one argument  "
        s = 1
    }
}       " end of methods "
```

図６．サンプルプログラム

表1．主な特権命令と実行マシンサイクル数

命　令		内　容　(ﾏｼﾝｻｲｸﾙ数)	
rcv	so, tfe	ﾒｯｾｰｼﾞ実行	(43)
svCnt	des	ｺﾝﾃｸｽﾄ退避	(46)
		再退避	(16)
run	so, tfe	ﾒｯｾｰｼﾞ実行再開	(14)
		ｺﾝﾃｸｽﾄ実行再開	(25)
crtCnt	des, so	ｺﾝﾃｸｽﾄ生成	(24)
slfCall	#, des, sel, so, ...	ｼｽﾃﾑｺｰﾙ	(50+8×#)
slfRet	so	ﾘﾀｰﾝ	(30)
mvID	des, so	des間接ﾃﾞｰﾀ転送	(14)
mvIS	des, so	so 間接ﾃﾞｰﾀ転送	(13)

so : ｿｰｽ　　# : 引数の数
des : ﾃﾞｽﾃｨﾈｰｼｮﾝ　　tfe : ﾃﾝﾎﾟﾗﾘﾌﾚｰﾑｴﾝﾄﾘ

表2．引数なしﾒｯｾｰｼﾞの受理処理の手数
(ﾏｼﾝｻｲｸﾙ)

	第1版	第2版
前処理	3217	1862
後処理	489	862
合　計	3706	2724

1ﾏｼﾝｻｲｸﾙ ＝ 125ns

ージの実行（rcv）である．また，これに伴いメッセージ／コンテクストの実行・再開（run）命令は，ユーザ用特殊レジスタに実行イメージが載っているときと，いないときで１１マシンサイクル異なる．

最も簡単な引数無しメッセージ m0 を受理するときのマシンサイクル数は表２のようになった．

ここで，前処理とはメッセージ割り込みを受け付けてから m0 を起動するまで，後処理とは m0 の終了後，メッセージ割り込みの終了するまでを指す．この結果から，ユーザメソッドを起動するまでの時間は約40% 短縮されたことが分かる．このことは，次の観点から重要である．すなわち，前処理が長くなった場合，その間に次のメッセージが到着する可能性も大きくなり，その場合，ユーザの実行は更に次の到着メッセージの割り込み前処理分待たされることになるからである（図1参照）．後処理が重くなったのは，メッセージレシーブ命令の実現に伴い，メッセージキュー管理が複雑になったことによる．合計の手数は，約2

表3．引数１つのﾒｯｾｰｼﾞ受理処理の手数
(ﾏｼﾝｻｲｸﾙ)

引数の種類	第1版	第2版
整数　0	4436	3225
文字列 'string'	5409	4556
配列　[1,2,3]	7949	5039
リスト {1,2,3}	8352	6123
辞書 <<<1;2,3;4,5;6>>>	13953	7098

文字列は，６文字
配列，リストは整数 1,2,3 の３要素を持つ
辞書は，(ｷｰ 1, 値 2), (ｷｰ 3, 値 4), ‥

5% 減少している．

次に，引数を１つ持つメッセージ m1: の受理にかかる手数の比較を表２に示す．引数の設定は，前処理に含まれ，後処理の手数は，上記のものと大きく変わらないため，ここでは合計のマシンサイクル数を示す．

この例では，約17～49% の高速化が図られている．一般に，引数の構造が複雑になるほど処理時間短縮の効果が大きい．

４．２　ユーザプログラムを用いた評価

ＯＳ第２版を用いていくつかの A-NETL プログラムのシミュレーションを行い，メッセージ当りのユーザの処理粒度，ユーザとシステムの実行バランス，オブジェクトサイズとローカルメモリ容量の関係，などを調べた．シミュレータは SONY NEWS 上にインプリメントされており，ＰＥとルータをそれぞれ１つのプロセスとして実現し，メッセージパッシングにはソケットによるプロセス間通信を用いている．使用したサンプルプログラムは，次の６種類７つである．

①ホップフィールドモデルのニューラルネットを用いた巡回セールスマン問題（５都市）
②３辺に５色の内，１色のついた９つの三角形ピースを隣合う色が同じになるように並べて大きな三角形を作る色合わせパズル
③原子の結合状態の変化による化学反応のシミュレーションについて，次の２通り
(1) $CO_2 + H_2 \rightarrow CO + H_2O$
(2) $C_3H_8 \rightarrow C_3H_6 + H_2$

④８つの２分木の葉にデータを格納し，これらを並列に検索・更新するデータベース

⑤22ゲートからなる３状態シーケンサ回路のシミュレーション

⑥行列式，逆行列，連立方程式の計算（実行時間は４×４行列の逆行列計算のもの）

実験に用いたＯＳ第２版は，メソッド総数 63，全体の大きさ約16.5KB で，第１版よりも 1.8KB ほど大きくなっている．

表４にシミュレーション結果を示す．

（１）処理の粒度

１つのメッセージを受理して起動されるユーザ処理粒度の平均は 453.2 マシンサイクルであった．これは，機械命令で20～30命令程度と非常に小さいものである．メッセージの持つ引数の数が，平均1.4個であることから，これらの受理に要するシステムの実行マシンサイクル数は，表３の値よりも若干大きなものとなり，4000 以上となる．従って，ユーザとシステムの実行時間の割合は 1:9.7 という比になる．このことは，細粒度のユーザ処理の間に約10倍のシステム実行が介在することを意味し，稼働率が上がっても，高並列とならない限り一般の問題において実行性能を期待することが難しくなる．

この関係を改善するためには，ユーザの粒度を大きくすることと，システムの粒度を小さくすることの両面からのアプローチが必要である．現在の実行モデルでは，ユーザのメソッド呼び出しは，同一ノード上での呼び出しを含めて，すべてキューを介したメッセージパッシングに帰着する．これに対して，ＯＳのメソッド呼び出しは，表１に示したようにスタックを用いたコール（slfCall）／リターン（slfRet）を用いて行われる．現在これをユーザに開放していないのは，セマンティクスの統一のためと，実行の安全性を保つためである．例えば，ＯＳのメソッドは動的に消去・変更されないが，ユーザのメソッドはその可能性がある．ただし，今後，各ノード上で基本的な機能を提供する算術関数や文字列操作などのライブラリオブジェクトをサポートする予定であり，これら同一ノード上でのＯＳを介在しないメソッド呼び出しも検討していきたい．

（２）アドレス化セレクタの効果

１オブジェクトに含まれる平均的なメソッド数は5.8個であり，メッセージセレクタのアドレス化は，2.9(=5.8/2)回の辞書探索を不要にしたといえる．

表4. シミュレーション結果

プログラム	ﾆｭｰﾗﾙﾈｯﾄ 巡回ｾｰﾙｽﾏﾝ	色合わせ パズル	化学反応の ｼﾐｭﾚｰｼｮﾝ1	化学反応の ｼﾐｭﾚｰｼｮﾝ2	2分木 ﾃﾞｰﾀﾍﾞｰｽ	ｼｰｹﾝｻ回路 ｼﾐｭﾚｰｼｮﾝ	行列計算	平均値
ｵﾌﾞｼﾞｪｸﾄ数	32	194	6	14	15	28	21	44.3
静的／動的	6／26	2／192	6／0	3／11	15／0	5／23	5／16	6／40.4
平均ｻｲｽﾞ（語）	297.7	343.2	187.3	210	209.1	67.9	163.8	211.3
平均ﾒｿｯﾄﾞ数	7.8	6.7	4	5.4	8.3	2.0	6.7	5.8
使用ＰＥ数	16	16	6	14	15	16	16	14.1
実行ﾏｼﾝｻｲｸﾙ								
ユーザ	3,566,293	2,728,298	129,351	342,880	116,728	61,003	72,715	1,002,467
システム	19,508,979	26,868,085	1,568,039	12,170,185	3,045,353	2,765,152	1,871,672	9,685,352
ﾕｰｻﾞ：ｼｽﾃﾑ	1：5.5	1：9.8	1：12.1	1：35.5	1：26.0	1：45.3	1：25.7	1：9.7
ＧＣ／ノード	1.5	2.6	0	1.4	0.1	0	0	
ﾒｯｾｰｼﾞ割り込み	2,588	4,305	289	2,024	558	582	255	1,514.4
サイズ（語）	8.2	6.1	6.7	5.9	5.6	5.68	6.45	6.38
引数の数	2.3	1.5	1.6	0.7	1.0	0.9	1.7	1.4
ユーザ／ﾒｯｾｰｼﾞ（ﾏｼﾝｻｲｸﾙ）	1378.0	683.2	342.9	169.4	209.2	104.8	285.2	453.2
未来トラップ	101	2,365	0	22	168	25	44	389.3
最大並列度	14	14	5	11	10	8	9	10.1
稼働率	4.4	8.0	0.9	5.2	1.9	4.1	2.1	3.33

ＰＥ１マシンサイクル ＝ １２５ｎｓ，　ローカルメモリ１語＝４０ビット

（３）ローカルメモリ容量

メモリ消費特性は問題の性質とアルゴリズムによって異なるが，実験に用いた規模のプログラムでは，ノード当りのＧＣ発生回数は０～４回であった．従って，寿命の長いコンテクストやリストなどのデータオブジェクトはほとんどなく，これらの構造体データによって消費される動的利用可能領域の大きさは十分であるといえる．また，オブジェクト割り付け領域 12K語には，平均的な大きさ211.3語のオブジェクトが５８個割り付け可能であると見積ることができる．この数は，A-ＮＥＴＬ による複数オブジェクトの同一ノードへの割り付け指定[1]に対しても十分なものであり，実装するメモリを減らした場合でも，実行モデルは損なわれないと考えられる．

（４）メッセージサイズとキューサイズ

平均的なメッセージサイズは 6.38 語で，その到着回数（メッセージ割り込み）は 1,514.4 回であった．このことは，4K語の循環入力メッセージキューを約2.4周したことを意味する．

（５）未来トラップ

未来トラップは，メッセージの返答の格納先に指定された変数を読み出そうとして，返答が返っていないときに起きる．このときＯＳの要するコンテクストチェンジの手数は約700マシンサイクルである．ユーザ処理の粒度が小さいと，メッセージ送信から返値の参照までの時間も短くなり，未来トラップも起こり易くなる．

（６）最大並列度

最大並列度は，初期化を除いて同時に実行が行われたノード数の最大値である．シミュレータが，クラスの存在するノードに動的オブジェクトを割り付けないなどの制約を考えると，ほぼ使用ＰＥ数に近い値が得られたといえる．

５．おわりに

A-ＮＥＴ ローカルＯＳ第２版の設計方針とシミュレータによる実験結果を中心に述べた．今回，言語処理系やシミュレータの制約から，インデックス付きオブジェクト[1]を用いたプログラムや１６ノードを越えるシミュレーションは行えなかった．また，ユーザとシステムの実行バランスについても，まだ満足のいく結果は得られていない．システムの実行時間を短縮するためには，メッセージキューなどのハードウェアサポートや実行モデルの簡単化が考えられる．しかし，単にシステムの実行時間を縮めても，その分ＰＥが遊んでしまっては意味がない．ルータとのバランスも考慮する必要がある．第３版では，これらの問題に取り組むとともに，データベースの検討なども行っていく予定である．

謝辞

本稿を書くにあたり，シミュレーション等に協力して頂いた大学院生の濱田君，および卒研生の浅利，片平，郡司，鈴木，佐々木，大谷の各君に感謝する．

参考文献

[1]T.Baba, et al.: "A Parallel Object-Oriented Total Architecture: A-NET," Proc. 1990 Int. Conf. on Supercomputing (to appear).

[2]寺岡 他："並列オブジェクト指向トータルアーキテクチャ A-ＮＥＴ －ＰＥのアーキテクチャー，" 情報処理学会第41回大会，6P-7(1990).

[3]茂木 他："並列オブジェクト指向トータルアーキテクチャ A-ＮＥＴ －ルータの設計－，" 情報処理学会第41回大会，6P-8 (1990).

[4]馬場 他："並列計算機におけるタスク割付けに関する一考察，" 電子情報通信学会コンピュータシステム研究会，CPSY 90-32 (1990).

[5]吉永 他："並列オブジェクト指向トータルアーキテクチャ A-ＮＥＴ のローカルＯＳ，" 情報処理学会オペレーティングシステム研究会，89-OS-45-5 (1989).

[6]濱田 他："並列オブジェクト指向トータルアーキテクチャ A-ＮＥＴ のプログラムデバッグ方式，" 情報処理学会第40回大会，1L-5(1990).

[7]T.Baba, et.al.: "A Network-Topology Independent Task Allocation Strategy for Parallel Computers," Proc. 1990 Int. Conf. on Supercomputing (to appear).