プログラミング－言語・基礎・実践－ 18－12
（1994. 7. 21）

共有メモリ型マルチプロセッサにおける並行オブジェクトの実行系の評価

青柳 洋一, 中林 嘉徳, 岩田 竜一, 上原 稔, 森 秀樹
東洋大学工学部情報工学科

スレッドまたはLWPをサポートした共有メモリ型マルチプロセッサマシンの普及に伴い、効率の良い並行オブジェクト指向言語の実行系への要求が高まっている。しかしマシンの性能にはバラつきがあり、効率の良い実装のためには、プロセスやスレッドなどの実行単位の組み合わせで考えられる多様な実行形態の中から、効率の良いオブジェクトの配分モデルの選択が必要である。本研究では、実行方式のモデル化をオブジェクト間の通信の結合度に応じたグループ分けと、実行単位の組み合わせで行い、その評価から得られた結果をもとに必要最小限のデータから最適な実行モデルを予測する。

Evaluation of Execution System for Concurrent Objects on Shared Memory Multiprocessors

Yoichi Aoyagi, Yoshinori Nakabayashi, Ryuichi Iwata, Minoru Uehara, Hideki Mori
Department of Information and Computer Sciences,
Toyo University

Recently, shared memory multiprocessors which support light weight processes like threads have been developed. At that same time, high performance executing system for concurrent objects is required. However, their performance are varied. In these macines, efficient execution is dependent on the combination of several execution units (e.g. processes, threads and simulated objects), object grouping, communication frequency, etc. In this paper, we propose how to predict an optimal executing system for concurrent objects on individual executing environments, by using simple evaluations.

1 はじめに

細粒度並行オブジェクト計算は、通信と通信の間隔が極めて短い細粒度オブジェクトの集合からなる。このような計算はオブジェクトの並列度が高く、超並列マシンでの効率良い実装に適しており、将来が有望である。本研究では、このような細粒度の並行オブジェクト指向言語実装の支援を目的としている。

並行オブジェクト指向言語がいくつか開発されている [1],[2],[3]。この中では、プロセスを用いてオブジェクトを実現したもの、スレッドを用いて実現したもの [1],[2]、仮想機械でオブジェクトをシミュレートしたもの [3] などがある。各方式は一長一短があり、またこれらの組合せも豊富である。このような状況で、並行オブジェクト指向言語においてオブジェクトの効率良い配置をした実装が要求され、必要最小限のデータから効率良いオブジェクトの実装法を選択する手段が求められている。

本稿ではアーキテクチャに応じて、効率の良いコードを生成するための、最適モデル選択法を提案する。

一般に、ある処理にかかる実行時間は、以下の式で与えられると考えられる。

(全処理時間)
=(通信時間) + (主処理時間) + (待ち時間)

主処理時間は、プロセッサへのオブジェクトの配分が等しければ、実行の形式が変わってもほぼ一定である。また、待ち時間は、通信と主処理との相関から決まる要素である。したがって、通信時間の分析を行うことにより、処理全体にかかる時間の最も短い実行方式の推測が可能と考えられる。

本論文で提案する方式では、必要最小限のデータを得て評価を行うことにより、アーキテクチャ、オブジェクトのモデル構成に応じた最も効率良い実行モデルの選択をする。

本論文は以下のような構成になっている。最初に、いくつか考えられるオブジェクトのモデル構成の中から代表的なものをあげ、評価モデルの構成について述べる。次に、必要最小限のデータから最適モデル選出をするアルゴリズムを提案し、通信所要時間の推測に用いる一次式を紹介する。そして、評価を行い、最後に結論を述べる。

2 実行形式のモデル化

並行オブジェクトはオブジェクトのネットワークで構成されるため、データの到着順の保証されるストリーム通信が望ましい。したがって、本論文ではストリーム通信をオブジェクトの通信手段として採用している。オブジェクトは通信の結合度に応じてグループ分けを行い、そのグループ構成に対する実行単位の割り付け方で、モデル化を行う。

2.1 ストリーム通信

オブジェクト間通信はメッセージ単位で行われる。多くのオブジェクトシステムではプログラミングの煩わしさを避けるため、メッセージの到着順を保証している。このような通信形態はストリームとして表現される。

ストリーム通信の特徴は以下の通りである。

- メッセージが連続したデータの流れとして扱われるため、メッセージの到着順が保証される。
- バッファリングにより非同期通信が容易に実現され、バッファサイズで通信の量子を調整できる。また、大量の情報を送受するのに適する。
- UNIX などの OS と親和性がよい。プロセス間通信は pipe で、スレッドや SO 間通信は共有変数で容易に実現できる。

2.2 実行単位

実行モデルを形成する、異粒度の実行単位として、プロセス、スレッド、Simulated Object(以下、SO) の 3 つの代表的な基本実行系を用いた。これらの特徴を以下に示す。

- **プロセス**
  UNIX のプロセス。全モデルにおいて全体を統括するプロセスが必ず 1 つ存在する。ストリーム通信の手段として pipe システムコールで通信。pipe は、socket によって実現されており通信コストが大きい。また、プロセススイッチング等のコストも他の実行単位に対して大きい。

- スレッド
  Mach cthread。mutex によるロックを用いた共有変数でストリーム通信を実現。mutex による排他制御を行いながら共有変数へのアクセスを行うため、SO の通信と比較してその通信コストは大きい。プロセスよりスイッチングコストは小さい。

- **Simulated Object**(SO)
  上位の実行単位によって、並行にシミュレート実行されるデータオブジェクト。共有変数でストリーム通信を実現するが、共有変数への同時アクセスは無いので排他制御を簡素化できる。そのため通信コストは最小である。また、シミュレート実行のためスイッチングコストは最も小さい。ただし並列性はない。

これらの特徴を表 1にまとめる。なお、表中の他コストとは、スイッチングコストなどを指す。また、本文中の表、グラフでは、プロセス、スレッドに対して Prc,Thr の略称を用いる。

実行単位	通信手段	通信コスト	他コスト
プロセス	pipe	大	大
スレッド	共有変数,lock	中	中
SO	共有変数	小	小

表 1: 各実行単位の特徴

2.3 実行方式のモデル化

オブジェクト間の通信の結合度の強いものを同一グループとし、各グループにおける処理機能の大きさが均等になるようにオブジェクトを振り分けてグルーピングを行う。そのグループ構成に対する実行方式の割り付け方で、表 2に示す 6 モデルに分類した (図 1参照)。ここで、グループとオブジェクトが同じ実行単位になるものは (1),(2),(3) に含めた。

3 最適モデル推測のアルゴリズム

処理全体にかかる予測時間 Tは、摸式的な式 (1)

$$T = T_{com} + T_{main}/N_{div} + T_{wait} \qquad (1)$$

全体	グループ	単体	
プロセス	—	プロセス	(1)Prc × n
		スレッド	(2)Thr × n
		←SO	(3)SO × n
	プロセス	スレッド	(4)P(T × n) × m
		←SO	(5)P(S × n) × m
	スレッド	←SO	(6)T(S × n) × m

※ '←' は、SO が上位の実行単位によってシミュレート実行されることを示す

表 2: 実行形式の代表的な組合せ例

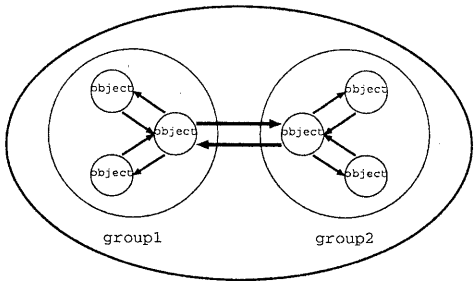

図 1: モデル構成例

T_{com} : 通信時間の予測
T_{main} : プロセッサ振り分けを考慮しない主処理時間の予測
T_{wait} : 待ち時間の予測
N_{div} : プロセッサへの負荷の振り分け方で決まる除数。

で与えられると予想できる。このうちの、通信所要時間 T_{com} を求めるための手段として、式 (2) を提案する。

$$T_{com} = f(n_1, n_2, n_3; C_1, C_2, C_3)$$
$$= C_1 * n_1 + C_2 * n_2 + C_3 * n_3 \qquad (2)$$

T_{com} : 通信時間
C_i : 各実行単位の通信コスト (1 回当たりの通信時間); 個々のマシン固有のパラメータ
n_i : 各実行単位の通信回数 (問題毎に異る)

本研究では、実際の所要時間ではなく、各モデル間の処理時間の順序関係を求めることを目的としているので、n_1, n_2, n_3の値は実際の通信回数ではなく、全通信回数に対する各実行単位の通信回数の割合で

もよい。

ここで、アルゴリズムの概要をまとめると以下のようになる。

1. オブジェクトの実装を行うマシンで、評価プログラムにより通信コスト C_i を計算する。
2. 予測式 T_{com} を導く。
3. 評価対象のプログラムを解析して通信回数 n_i を求める。
4. 得られたパラメータをもとに、評価対象の通信所要時間を推測し、その値の最小となるモデルを選択する。

4 評価

4.1 評価モデル

次のようにシンプルで、データ解析のしやすい問題について実行時間の評価を行った。

1. A 支社、B 支社の 2 つの支社がある。
2. A 支社には 1 台のデータベースサーバ (以下、SA) と、2 台のクライアント (以下、CA1,CA2) があり、クライアントは SA にデータを要求して作業を行う。
3. A 支社のクライアントが B 支社のデータを必要とするときは、SA を介してデータを受け取る。
4. B 支社についても同様の構成で、SB,CB1,CB2 が存在する。
5. 支社 A,B をグループとし、サーバ、クライアントをオブジェクトとしてモデルを形成した。

クライアントの数を増やした場合、支社の数を 4 つに増やした場合についても評価を行った。

4.2 評価

本論文では、OMRON LUNA88K(1/2/4 プロセッサ) 上における評価を行った。上記評価モデルについて条件 (プロセッサ数、グループ構成、通信回数) を変え、処理時間の計測を行い、通信所要時間を求めた。計測はできるだけ正確な値を求めるべく、シングルユーザモードで行った。

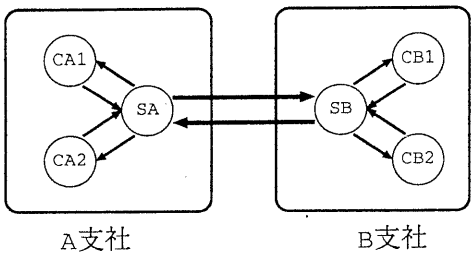

図 2: 評価の基本モデル

式 (2) より、1,2,4 プロセッサにおける通信所要時間 f_1,f_2,f_4はそれぞれ、以下の式で表される。添え字 1,2,4 はプロセッサ数に応じた固有の値を示す。

$$f_1 = C_{p1} * n_p + C_{t1} * n_t + C_{s1} * n_s \quad (3)$$

$$f_2 = (C_{p2}/N_{pdiv2})*n_p+(C_{t2}/N_{tdiv2})*n_t+(C_{s2}/N_{sdiv2})*n_s \quad (4)$$

$$f_4 = (C_{p4}/N_{pdiv4})*n_p+(C_{t4}/N_{tdiv4})*n_t+(C_{s4}/N_{sdiv4})*n_s \quad (5)$$

C_{pi} : プロセス間の通信コスト
C_{ti} : スレッド間の通信コスト
C_{si} :SO 間の通信コスト
n_p, n_t, n_s : プロセス、スレッド、SO の間の通信回数 (比)
$N_{*div\ i}$: 負荷のプロセッサへの分散によって変わる除数

4.2.1 1 プロセッサにおける評価

計測により、1 プロセッサにおける各実行単位の 1 通信の所要時間を得た。式 (3) において各コストは、

$C_{p1} = 0.55[ms]$
$C_{t1} = 0.3[ms]$
$C_{s1} = 0.0235[ms]$

となった。この値を式 (3) に代入して式 (6) が得られる。

$$f_1 = 0.55 * n_p + 0.3 * n_t + 0.0235 * n_s \quad (6)$$

計算例

評価モデルの (2 グループ) × (3 オブジェクト) の場合において、グループ間、オブジェクト間の通信回数 $N_{Group} = 2 * 10^3$(回)、$N_{Obj} = 8 * 10^3$(回) を与えたときの最適形態を選ぶ。式 (6) に従って各モデルの通信時間を求めると、表 3 となる。[1]

この結果から、この場合、"SO のみ" で実現したモデルが最も効率の良いことが分かる。予測値と値の離れたものもあるが、6 モデル内の順序関係は変わらないので順序の評価は有効である。このときの処理時間のグラフを図 3 に示す。

モデル構成	計算値	測定値	順位
6SO	0.235	0.232	1
2Thr × 3SO	0.788	0.662	2
2Prc × 3SO	1.29	1.39	3
6Thr	3.3	2.96	4
2prc × 3Thr	3.5	3.72	5
6Prc	5.5	5.43	6

※ グループ間 2 通信、オブジェクト間 8 通信

表 3: (2 グループ) × (3 オブジェクト) における通信所要時間 [sec](1 プロセッサ)

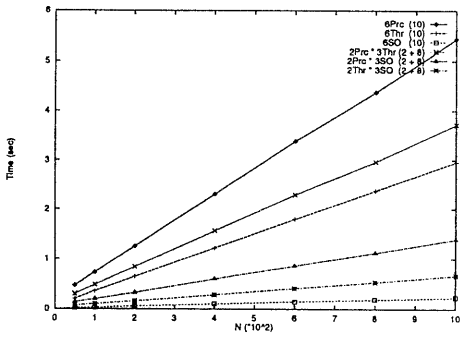

図 3: (2 グループ) × (3 オブジェクト) の実行時間 (1CPU)

[1]グループとオブジェクトの実行単位が同じものは、(2Prc × 3Prc) → (6Prc) のように表している。

4.2.2 2 プロセッサにおける評価

式 (4) において、各通信コストの値は以下のようになった。

$C_{p2} = 0.78$ $[ms]$
$C_{t2} = 0.19$ $[ms]$
$C_{s2} = 0.0235$ $[ms]$

また、除数の値は各モデルで以下のように近似できることが分かった。

モデル	N_{pdiv2}	N_{tdiv2}	N_{sdiv2}
SO のみ	—	—	1
Thr× SO	—	2	2
その他	1	1	2

これらの値を式 (4) に代入すると、以下の式が得られる。

$$f_2 = 0.78 * n_p + 0.19 * n_t + 0.0235 * n_s/2 \quad (7)$$

$$f_2' = 0.0235 * n_s \quad (8)$$

$$f_2'' = (0.19 * n_t + 0.0235 * n_s)/2 \quad (9)$$

ここで、実測値において 2 グループと 4 グループとで、通信時間の順位の逆転が起きるが (表 4.2.2中 1 位と 2 位、3 位と 4 位; 図 4,5)、この逆転は上式 (7),(8),(9) で予測が可能である。ただし、オブジェクト数が少ないときには予測値のわずかなずれによって、順位の予測が外れることがある。ただし、このような場合、どちらを選んでもあまり重要な差はない。

この結果に従えば

- 2 グループでモデル構成をするならば、(2Thr) × (nSO)
- グループ数が 4 以上ならば、SO のみ

で実装を行うと効率が良いことになる。

順位	2 グループ	4 グループ
1	Thr × SO	SO × SO
2	SO × SO	Thr × SO
3	Prc × SO	Thr × Thr
4	Thr × Thr	Prc × SO
5	Prc × Thr	Prc × Thr
6	Prc × Prc	Prc × Prc

表 4: グループ数による順位の変動 (2 プロセッサ)

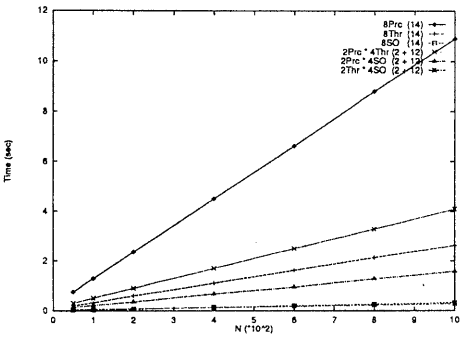

図 4: 2 プロセッサにおける順位の変動 (A); 2 グループ×4 オブジェクト (2 CPUs)

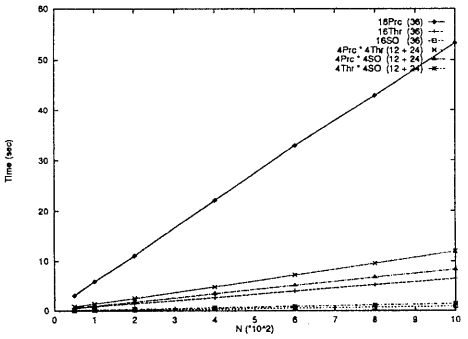

図 5: 2 プロセッサにおける順位の変動 (B); 4 グループ×4 オブジェクト (2 CPUs)

4.2.3 4 プロセッサにおける評価

式 (5) において、通信コスト、除数は以下のようになった。

$C_{p4} = 0.8 \quad [ms]$

$C_{t1} = 0.15 \quad [ms]$

$C_{s1} = 0.0235 \quad [ms]$

モデル	N_{pdiv2}	N_{tdiv2}	N_{sdiv2}
SO のみ	−	−	1
2Thr × SO	−	2	2
4Thr × SO	−	4	4
その他	1	1	4

これらの値を式 (5) に代入すると、式 (10)〜(13) が得られる。

$$f_4 = 0.8 * n_p + 0.15 * n_t + (0.0235 * n_s)/4 \quad (10)$$

$$f_4' = 0.0235 * n_s \quad (11)$$

$$f_4'' = (0.15 * n_t + 0.0235 * n_s)/4 \quad (12)$$

$$f_4''' = (0.15 * n_t + 0.0235 * n_s)/2 \quad (13)$$

ここでも、やはり 2 グループ、4 グループの間で順位の変動 (表 4.2.3中 3 位と 4 位; 図 6,7) が起きており、式 (10)〜(13) で予測が可能である。2 プロセッサの場合と同様、オブジェクト数が少ない場合に予測値のわずかなずれで順位の予測が外れることもあるが、微妙な差であり、あまり重要なことではない。

順位	2 グループ	4 グループ
1	Thr × SO	Thr × SO
2	SO × SO	SO × SO
3	Prc × SO	Thr × Thr
4	Thr × Thr	Prc × SO
5	Prc × Thr	Prc × Thr
6	Prc × Prc	Prc × Thr

表 5: グループ数による順位の変動 (4 プロセッサ)

4.3 1,2,4 プロセッサの比較

- プロセッサ数を変えていった中で、常に効率の良いのは、"SO のみ" のモデルと "Thr × SO" モデルであることがわかる。つまり、プロセスを用いるのは、その必然性があるとき[2]や、スレッドを実装しないシステムに限られることになる。
- SO 方式のみでモデルを形成したモデルはプロセッサ数を変えても通信時間は変化がなかった。
- オブジェクトを全てプロセスで実現したモデル (図 8) では、オブジェクト数が 16(36 通信/一巡) より上のものは、急激に処理時間が増大している。これは、システムが持つバッファの上限を越えたためと考えられる。

[2]例えば、メモリ空間やポートなどの資源を新たに割り当てたいとき等。

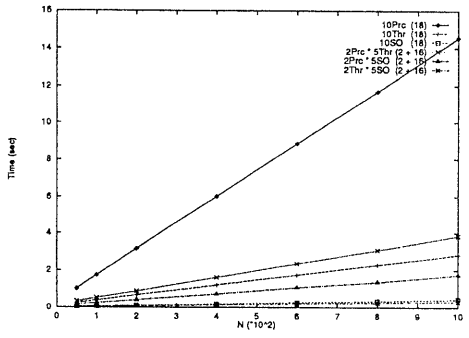

図 6: 4 プロセッサにおける順位の変動 (A); 2 グループ×5 オブジェクト (4 CPUs)

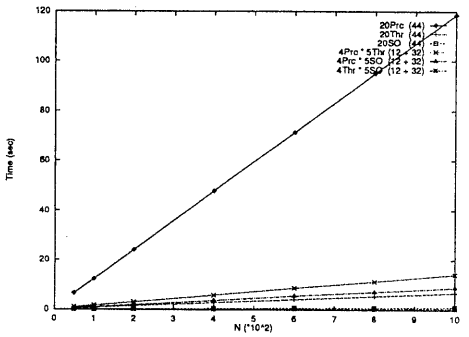

図 7: 4 プロセッサにおける順位の変動 (B); 4 グループ×5 オブジェクト (4 CPUs)

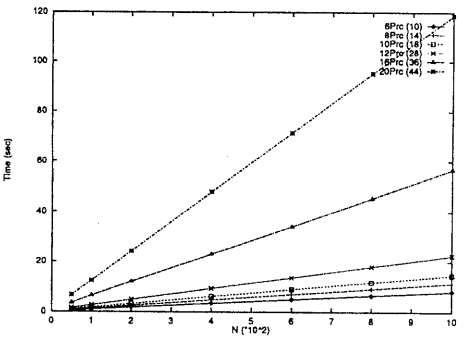

図 8: オブジェクトは全てプロセス (4 CPUs)

- プロセス×SO のモデルでは、4 グループ (4 プロセス) にしたとき、SO の数が増えても、処理時間はほとんど変わらなくなる (図 9)。プロセスの負荷に対して、SO の負荷が軽すぎるためである。
- マルチプロセッサシステムにおいて、Thr × SO のモデルが他の構成のモデルよりも効率が良い、ということの確認ができた。

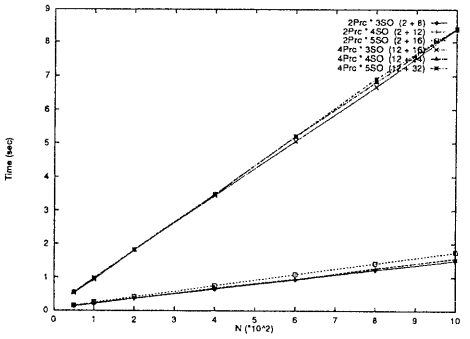

図 9: プロセス×SO(2 CPUs)

一般に、通信処理時間は、以下の式で求まる。

- SO のみからなるモデル

$$f_1 = f_2 = f_4 = C_s * n_s \qquad (14)$$

- Thr × SO モデル

$$f_i = (C_{ti} * n_t + C_s * n_s)/N_{div\ i} \qquad (15)$$

i はプロセッサ数

- その他のモデル

$$f_i = C_{pi} * n_p + C_{ti} * n_t + (C_{si}/N_{div\ i}) * n_s \quad (16)$$

$C_{p\ i}, C_{t\ i}, C_s$: プロセス, スレッド,SO の通信コスト (C_p, C_tはプロセッサ数に応じて変動する)
n_p, n_t, n_s : プロセス, スレッド,SO の通信回数 (比)
$N_{div\ i} : Minimum(N_{CPU}, N_{Group})$
i はプロセッサ数

5 まとめ

以上の評価によって得られた一次式に条件(グループの配分、プロセッサ数)とパラメータ(グループ間通信数、オブジェクト間通信数)を与えることにより、どのモデルの処理が最も効率良いか、おおよその予測が可能となった。

もっと複雑なモデル構成の場合は、本稿でとりあげた式を組み合わせることにより、評価が可能である。

我々が開発中のシステムでは、ある目的の処理に対していくつかの実行形式の出力のバリエーションが選択可能である。このシステムに本研究の評価を適用することにより、効率良い実装が可能となる。

予測では、なるべく少ない情報からできる限り実測値に近い値を導くのがよい。問題はあらかじめ与えられる情報量と実測値との誤差の大きさである。誤差が小さくとも情報量が多ければ予測には適さない。そこで本論文は必要最小限の通信コストから全体の反応時間の予測を行う。このとき重要なのは順序関係である。

今後、さらに複雑な評価モデルや、負荷をかけた場合について、より正確な予測を目指していく。

参考文献

[1] OMRON. LUNA-88K UniOS-Mach 解説書.

[2] R. Trehan, N. Sawashima, A. Morishita, I. Tomoda, A. Inoue, and K. Maeda. Concurrent Object Oriented C (cooC). *ACM SIGPLAN NOTICES*, 1993. February.

[3] Yasuhiko Yokote. *The Design and Implementation of ConcurrentSmalltalk*. World Scientific, 1990.