アルゴリズム 3－1
（1988. 9. 22）

並列アルゴリズムに対する アルゴリズム・アニメーション

油川達昭，　榎原博之，　中野秀男，　中西義郎
大阪大学工学部通信工学科

最近アルゴリズムの研究、開発において注目されてきている“アルゴリズム・アニメーション”に注目し、この技法の並列計算のシミュレーションへの適用を考察している。“アルゴリズム・アニメーション”は、アルゴリズムの振舞いの過程を逐次画面上に図的に表示して、その流れをより把握しようとするものである。具体例として、“ソーティング問題”と“最大値問題”を取り上げ、ＵＮＩＸ４．２ＢＳＤ上で作成した並列計算シミュレータとＸウィンドウ・システムを用いて、問題に対する並列アルゴリズムのシミュレーションの視覚化を実現している。そして、この技法がアルゴリズムの振舞いの把握を容易にすることを確かめている。

ALGORITHM ANIMATION FOR PARALLEL ALGORITHMS

Tatsuaki YUKAWA, Hiroyuki EBARA, Hideo NAKANO and Yoshiro NAKANISHI
Department of Communication Engineering
Faculty of Engineering, Osaka University
2-1 Yamada-oka, Suita, Osaka, 565 Japan

Recent words on the "Algorithm animation" technique suggests that this technique is a powerful tool in the research and development of algorithm. This note is an application of this technique to the simulation on parallel computing. For "the sorting problem" and "the maximum integer problem", we develop an animation of parallel computing, using the X-window system and a parallel simulator constructed on UNIX4.2BSD. Through this experiments, we appeal the possibility and useful of this technique.

1　はじめに

最近、アルゴリズムの研究、開発を支援する有力な技法として“アルゴリズム・アニメーション”が注目されてきている[3,4]。

この技法は、紙面に記述された形態では、アルゴリズムの振舞いの全体的な流れを把握するのが容易ではないことに対する１つの対策であり、アルゴリズムの振舞いの過程を逐次画面上に図的に表示して、その流れをより把握しやすくしようとするものである。

また、計算の分野では、並列計算に関する研究が着実に進展してきており[5]、並列計算機とともに並列計算機上で動く並列アルゴリズムの研究、開発が重要になってきている[1]。

並列計算では、並列計算機のアーキテクチャと並列アルゴリズムとを関連させて扱わなければならないため、その研究ではしばしばシミュレータを使う[2]。その際並列アルゴリズムの振舞い（ 逐次アルゴリズムに比べて把握しにくい ）の把握を容易にするには、アルゴリズム・アニメーションの技法が有力と考えられる。

本報告では、ある並列計算機アーキテクチャ上での並列アルゴリズムのシミュレーション実験に際して、シミュレーション過程を視覚化することを課題にして、ＵＮＩＸ４．２ＢＳＤ上で作成した並列計算シミュレータとＸウィンドウ・システムを用いた視覚化を試みている。具体的には、この試みを“ソーティング問題”と“最大値問題”とに対する並列計算に対して実現し、実現にあたって考慮すべき点並びにアルゴリズムの振舞い把握に対する有効性を検討している。

2　並列計算機のシミュレータ

2．1　シミュレータ

並列計算では、与えられた問題に対して最適なアーキテクチャを持った計算機上での、最適な並列アルゴリズムを考えることが理想的である。従って、その研究では、並列計算機のアーキテクチャと並列計算アルゴリズムが問題になる。

並列計算のシミュレーション実験のためのシミュレータの構成で、以下のような柔軟性を持たせておくことが望ましい。

１）　任意なアーキテクチャを構成することができる。
２）　メモリ構成は、ローカルメモリ型、共有メモリ型、ローカルメモリ型と共有メモリ型の複合型等任意にできる。
３）　プロセッサ間の接続は、高速バス、メモリスワップ等任意のものに対応できる。
４）　分散型の並列計算機にも適用できる。

ここでは以上のことを考慮に入れて、ＵＮＩＸ４．２ＢＳＤ上に１つの並列計算機のシミュレータを作成する。このシミュレータでは、並列計算機での１つのプロセッサを１つのプロセスを対応させ、プロセッサ間の通信は、ＵＮＩＸ４．２ＢＳＤ上のプロセス間通信の手段である“ソケット”の機能を利用し、“データマネージャ”を用いて間接的に行なうようにしている。このような方式をとった理由と、その機能は次項で述べる。

2．2　データマネージャ

並列計算機の効率は、対処する問題にもよるが、普通個々のプロセッサにおけるアルゴリズムの効率だけではなく、プロセッサ間の通信量が大きな比重を占める。従って、シミュレータの構成にあたっては、プロセス間通信の状況把握が容易に行えることが望ましい。

ここで採用したデータマネージャを用いた間接的な通信方式、つまり送信側のプロセスは送りたいデータを一旦データ通信管理用のプロセス“データマネージャ”に送り、受信側のプロセスはデータマネージャにデータを読み取りにいく、という方法（ 図１参照 ）は、以下の利点を考慮にいれてのものである。

１）　プロセス間通信の状況把握が容易
　通信を直接行う方式では、状況把握は全てのプロセスを調べなければならなくなり、やっか

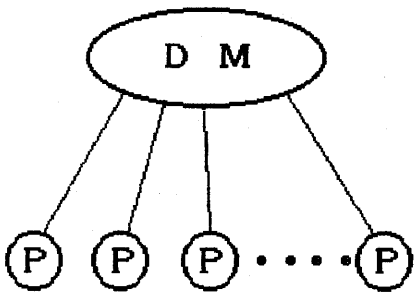
いである。これに対して間接的な方式では、通信はデータマネージャにおいて１元的に管理されることになり、状況把握はデータマネージャを調べるだけですみ、容易に行える。

２）　データ伝送路の数が少ない

直接的な通信では、もし全てのプロセス間に伝送路を設けなければならないとすると、伝送路の数はプロセス数に対して２次関数的に増加することになる。それに対して間接的な通信では、個々のプロセスとデータマネージャを接続すればよいので、伝送路の数は線形で押さえることができる。

３）　並列処理の過程の把握が容易

並列処理の過程を視覚化するにあたって、振舞いの状況把握は主にデータマネージャを調べるだけでよく、容易に行える。



ＤＭ：データマネージャ
Ｐ　：プロセッサ

図１　データマネージャ

２．３　アルゴリズムの視覚化

シミュレータ上での並列アルゴリズムの振舞いを視覚で観察できるようにする“アルゴリズムの視覚化”にはＸウィンドウ・システムを採用する。Ｘウィンドウ・システムは典型的なサーバ・クライアント方式のマルチウィンドウ・システムで、ディスプレイ装置に対する管理と操作を集中的に行なうＸサーバ・プロセスとＸサーバに要求を出すクライアント・プロセス（ユーザ・アプリケーション）の組合せで機能する。加えて、このシステムは、ウィンドウのオーバラップを可能にし、ビットマップ・ディスプレイ上で機能するという特徴を持っている。

Ｘウィンドウ・システムを採用した理由は、主として以下に示すような点で表現の自由度が高められることによる。

１）　ビットマップ・ディスプレイ上で機能するので、グラフィック表現に関して有用である。

２）　任意の位置にウィンドウを開くことができ、かつオーバラップ形のウィンドウ・システムであるので、ウィンドウの数が多くなってもその大きさを縮小する必要がない。

Ｘウィンドウ・システムを使っての視覚化にあたって、次に挙げるような機能を実現している。

１）　アーキテクチャの描画機能

この機能のもとで、シミュレートする並列計算機のアーキテクチャが、最低限の情報を与えるだけで簡単に表示できる。例えば、鎖状構造ではプロセッサ数を指定するだけで、また２進木構造では、その深さを指定するだけで描画できるようにしている。

２）　プロセッサ状態の表示機能

この機能のもとで、プロセッサが現在稼動しているかどうか、どの様な仕事をしているのかが図的に表示できる。例えば、稼動しているプロセッサは黒く、稼動していなければ白く表示できる。

３）　各プロセッサ内状態の個別表示機能

この機能のもとで、プロセッサ内の状態（あるデータの値等）がプロセッサ別に表示できる。つまり、各プロセッサが保持しているデータの値などを、それぞれ対応するウィンドウに表示できる。

４）　プロセッサ間の通信表示機能

この機能のもとで、あるプロセッサが送信可能であるか、受信可能であるか、あるいは実際に通信中であるのかなどが表示できる。例えば、プロセッサ間で通信が行なわれていることが、プロセッサ間の矢印で表示できる。

5） シミュレーションの中断，続行及び中止機能

この機能のもとで、ウィンドウ上に中断，続行及び中止を指示する欄を設け、それをマウスでクリックすることによって、シミュレーションを一時中断し、その後再開したり、また、シミュレーションを中止したりすることができる。

3 適用例

前章までに述べた視覚化の技法を、“ソーティング問題”及び“最大値問題”に対して適用してみた。以下に、その実験について述べる。

3．1 ソーティング問題

n(=r•k)個の自然数を非減少順でソートする問題を考える。ソーティング・アルゴリズムとしては、bubble sortを並列化した odd-even transposition sort を変形したものを[1]、また、このアルゴリズムに対する並列計算機のアーキテクチャとしては、ローカルメモリ M_i(i=1,2,3,･･･,k)を持ったプロセッサ P_iが k個鎖状に結合されたものを用いる。（ 図2参照 ）

P：プロセッサ

図2 鎖状構造

3．1．1 ソーティング問題の並列アルゴリズム

以下、プロセッサ P_iにおけるアルゴリズムを示す。

初期状態： 各プロセッサ P_iは r個のデータを M_iに記憶している。

step 1

各 P_iは、M_iに記憶されている r個のデータをソートする。

step 2

for m=1 to k

m=oddの時 step 2.1を行なう。

m=evenの時 step 2.2を行なう。

step 2.1

以下の操作は P_{2x-1}(x=1,2,3,･･･,[k/2])が行なう。

step 2.1.1

M_{2x}に記憶しているデータを M_{2x-1}に転送する。

step 2.1.2

M_{2x-1}に記憶している 2r個のデータをマージソートする。

step 2.1.3

step 2.1.2で得られた結果の大きな数から r個を M_{2x}に送る。

step 2.2

以下の操作は P_{2x}(x=1,2,3,･･･,[(k-1)/2])が行なう。

step 2.2.1

M_{2x+1}に記憶しているデータを M_{2x}に転送する。

step 2.2.2

M_{2x}に記憶している 2r個のデータをマージソートする。

step 2.2.3

step 2.2.2で得られた結果の大きな数から r個を M_{2x+1}に送る。

3．1．2 アルゴリズムの視覚化

この問題についてのアルゴリズムの視覚化では、2．3節で述べた機能を以下のように使っている。（ 付図1参照 ）

1） アーキテクチャの描画機能

アーキテクチャが鎖状構造のものであるので、そのプロセッサ数を指定するだけで表示する。

2） プロセッサ状態の表示機能

プロセッサを□で表し（ 内側の数字はプロセッサ番号 ）、プロセッサが稼働している時は、■のようにプロセッサを黒く表示する。

3） 各プロセッサ内状態の個別表示機能
　各プロセッサが持っているデータをそれぞれ対応するウィンドウに表示する。
4） プロセッサ間の通信表示機能
　プロセッサ間で通信が行なわれている時には、プロセッサ間に矢印➡を表示する。
5） シミュレーションの中断・続行及び中止機能
　図の左下の "CONTINUE" "STOP" "KILL"のボタンのところでマウスボタンをクリックして、“シミュレーションを続行する” “シミュレーションを中断する” “シミュレーションを終了する”という操作を行なう。

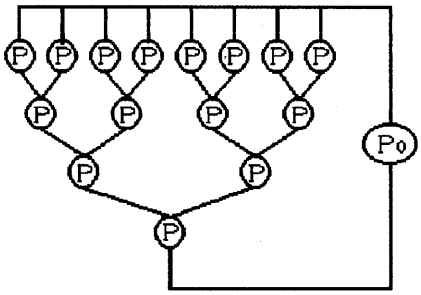
尚、この並列アルゴリズムは同期アルゴリズムであり、シミュレータでは、各プロセッサ間で通信を行ない同期をとる必要がある。従って、視覚化においても同期中を表示することが望ましい。このため、同期表示機能を実現し、同期中である時▌▌▌のように表示できるようにしている。この黒い棒は、各プロセッサが次のステップへの準備ができる毎に増加していき、全プロセッサ数が表示されると同期操作は終了する。（ 付図２参照 ）

３．２　最大値問題

ここで取り上げた“最大値問題”は、２進木の構造を利用して最大値を求めるアルゴリズムで解く問題である。このアルゴリズムに対する並列計算機のアーキテクチャとしては、ローカルメモリ M_i(i=1,2,・・・,k-1)を持ったプロセッサ P_iが２進木状に結合されたものを用い、さらにローカルメモリ M_0 を持った管理プロセッサ P_0が、２進木構造のrootプロセッサ及び全ての leafプロセッサに結合されているものとする。従って、２進木の深さがdであるとするとプロセッサの数は全部で $k=2^d$個必要である。（ 図３参照 ）

３．２．１　最大値問題の並列アルゴリズム

各 nodeプロセッサ P_i(i=1,2,・・・,k-1)におけるアルゴリズムは以下のようになる。



P　：プロセッサ
P_0：管理プロセッサ

図３　２進木構造

step 1
　子プロセッサから送られてきた２個のデータを M_iに記憶する。
　leafプロセッサの場合は管理プロセッサからデータを受け取る。
step 2
　２個のデータのうち大きな方を親プロセッサに送る。
　rootプロセッサの場合は管理プロセッサにデータを送る。
step 3
　もし、２個の子プロセッサからＥＮＤ信号を受け取ると、親プロセッサにＥＮＤ信号を送り終了する。

管理プロセッサ P_0におけるアルゴリズムは以下のようになる。

初期状態： M_0に n個のデータが記憶されている。

step 1
　$n=a(2^d-1)+b$となる a,bを求める。（ ただし、$b<2^d-1$ ）
step 2
　もし b=evenならば、十分小さな数をダミーとして１個データを追加する。

step 3

以下の操作を $ax2^{d-1}+[(b+1)/2]$回繰り返す。

step 3.1

leafプロセッサに各々２個ずつデータを送る。

もし、M_0にデータが２個以上存在しなければ、rootプロセッサから送られてきて、データが２個以上になるまで待つ。

step 3.2

rootプロセッサからデータが送られてきているか調べ、あれば M_0に記憶する。

step 4

leafプロセッサにＥＮＤ信号を送る。

step 5

rootプロセッサからデータが送られてくるのを待ち、来れば M_0に記憶する。（ これが最大値である。）

３．２．２　アルゴリズムの視覚化

この問題についてのアルゴリズムの視覚化では、２．３節で述べた機能を以下のように使っている。（ 付図３参照 ）

１）　アーキテクチャの描画機能

アーキテクチャが２進木構造のものであるので、その深さを指定するだけで描画する。

２）　プロセッサ状態の表示機能

nodeプロセッサを◎で表し（ 内側の数字はプロセッサ番号 ）、プロセッサが稼働している時は、●のように nodeプロセッサを黒く表示する。また、各 nodeプロセッサの両脇に表示されている数字は、子プロセッサまたは管理プロセッサから送られてきたデータである。nodeプロセッサの左にある数字は、左の子プロセッサから送られてきたものであり、右にある数字は、右の子プロセッサから送られてきたものである。ただし、管理プロセッサは表示されていない。

３）　プロセッサ間の通信表示機能

nodeプロセッサ間の通信が行なわれている時は、nodeプロセッサ間に矢印↓を表示し、子プロセッサが送信できる状態になっている時には、⊻を表示し、親プロセッサが受信できる状態になっている時には、⋀を表示する。

４）　シミュレーションの中断・続行及び中止機能

ソーティング問題の時と同様、図の左下に"CONTINUE" "STOP" "KILL"のボタンを設ける。

４　おわりに　ー　実験の評価

アルゴリズム・アニメーション技法に注目し、“ソーティング問題”と“最大値問題”を具体例として取り上げ、並列計算機上での並列アルゴリズムのシミュレーションの視覚化に関し、そのためのいくつかの機能を実現し、視覚化を試みてみた。

取り上げた具体例に関する限り、アルゴリズムの振舞いを把握できるだけのものにはできた。この例では、振舞いの把握によって、特に並列アルゴリズムの不良点や改良点が見い出せたということはなかったが、少なくとも振舞いの把握を容易にすることは確実である。今後、いろいろな並列アルゴリズムに適用してみていくことで、この技法がアルゴリズム研究、開発の支援技法として評価できるだけのものをもたらす可能性があると考えている。尚、視覚化の技法としては、さらに以下のようなことを考慮する必要がある。

１）　視認性について

見ている人に対して表示速度が適正かどうかという点に関して、プロセッサ間の通信の表示等について工夫、調整する必要がある。

また、プロセッサ数が多くなったり、結合が複雑になる場合（ 例えば高次元での結合のような場合 ）での表示も工夫する必要がある。このような場合、別のウィンドウに拡大部分を描画させるなどの対応策が考えられる。

２）　他のアルゴリズムの組み込みについて

他のアルゴリズムをシミュレータへ組み込む場合、グラフィックの部分に関する以外は、モジュール化が進んでいるので、そう困難なことはない。グラフィックの部分に関しては、１）での視認性ということも考慮すると、ある程度

アルゴリスムごとにその特徴を出せるように、その度ごとに開発する必要がある。従って、アーキテクチャの描画など基礎的な部分は、できる限り簡略化する必要がある。

謝辞　本研究を進める上で、協力いただいた東洋情報システム㈱木村幸敏氏に感謝いたします。

[参考文献]

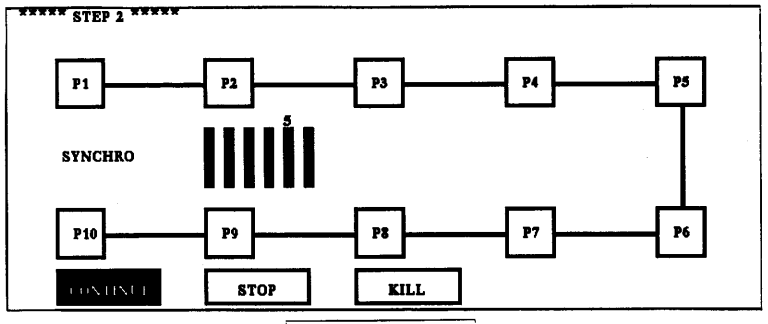
(1)Gerard Baudet & David Stevenson : "Optimal Sorting Algorithms for Parallel Computers", IEEE Trans. Comput., Vol.C-27 No.1 pp.84-87, January.1978

(2)木村, 榎原, 中野, 中西 : "ＶＬＳＩの設計規則検証に対するパラレルシミュレータ", 進学技報 CAS87-208, pp.1-5, 1987

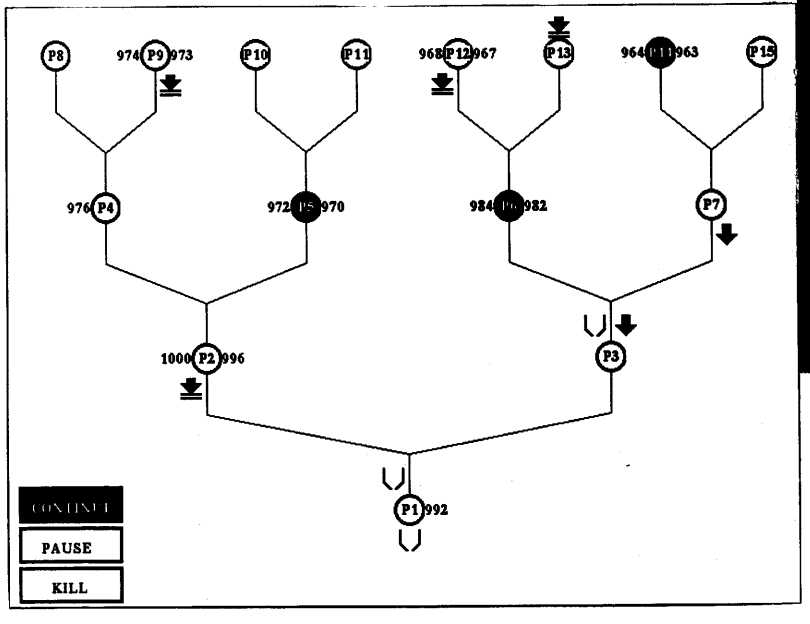
(3)Marc H.Brown : "Algorithm Animation", The MIT Press

(4)Marc H.Brown & Robert Sedgewick : "Techniques for Algorithm Animation", IEEE Software, Vol.2 No.1 pp.28-39, January.1985

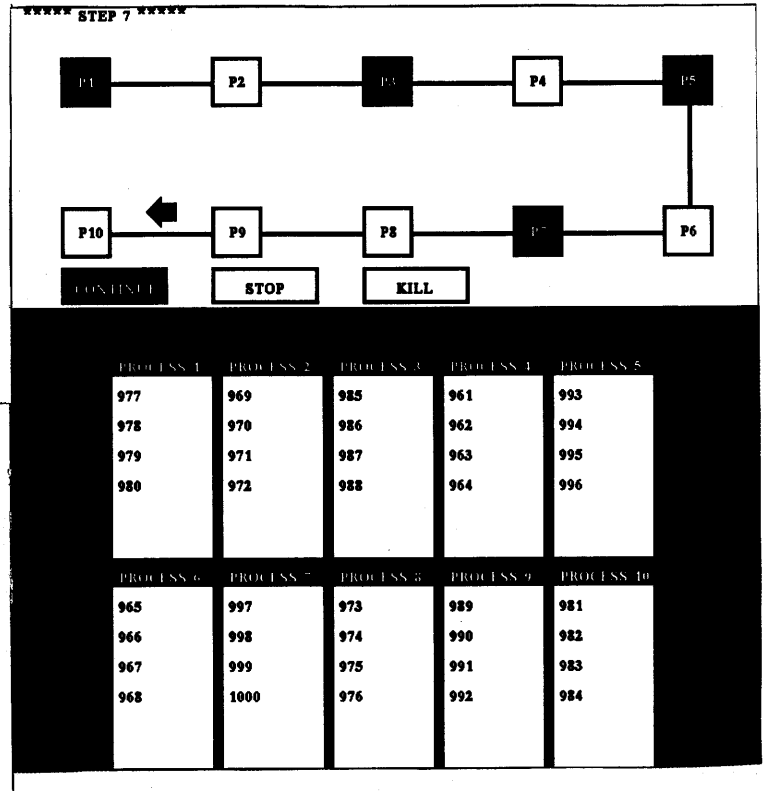
(5)高橋義造 : "並列処理のためのプロセッサ結合方式", 情報処理, Vol.23 No.3 pp.201-209, 1982



付図２　同期中



付図３　最大値問題



付図１　ソーティング問題