計算機アーキテクチャ 132−18
システムソフトウェアとオペレーティング・システム 80−18
ハイパフォーマンスコンピューティング 75−18
（1999. 3. 5）

低開発コストの分散計算方式の構築

○斉藤 賢二， 上原 均 (茨城大院)， 畠山 正行， 荒木 俊郎 (茨城大)
(ksaito, uehara, masayuki, araki)@cis.ibaraki.ac.jp

茨城大学工学部情報工学科, 〒316-8511, 日立市中成沢町 4-12-1

概要

コンピュータがネットワークに接続されて使われるのが常識となった現在においても，数値計算やシミュレーションにおける分散処理方式での「実利用率」は充分に低いのが現状である．我々は，その原因はドメインユーザの「開発に掛かる期間の長さと面倒くささ」にあると考えた．そこで，実行効率よりもドメインユーザが負担する「総合的な開発コスト」，即ち，分散システムの知識習得，開発期間，プログラミングの煩雑さ・簡単さ，等に重点を置き，既存のプログラムの分散化を行うシステムを構築した．そのためオブジェクト指向と C++を前提にした．その結果，分散共有配列 (DSA) を用いるシステム，分散プログラムを自動生成する ADS 共に従来から使われてきた PVM よりもかなり改善された良好な開発コストで実現されることが確認された．

A Distributied Computing Architecture for Low Cost Development

Kenji Saitou, Hitoshi Uehara, Masayuki Hatakeyama, and Toshirou Araki
Department of Computer and Information Sicences, Ibaraki University, Japan

Abstract

The present paper aims at presenting a distributied computing architecture for low cost development. The existing distributied computing architectures or sytems are very inconvenient and troublesome for the domain users to develop and use the distributed computing system without high development cost or time. We have newly developed two kinds of the distributed computing system that use a kind of the distributed shared array and a distributed computing program generator. Throughout the evaluation tests for the development period, development procedure complexity, and the execusion efficiency, our proposed destribution system hove shown the higher priority than the PVM.

1 はじめに

大規模或いは長大な計算時間の掛かる処理を分散して処理することにより短期間に結果を得，研究開発等を効率化することは以前から頻繁に行われてもおり，その技術も膨大な蓄積が行われてきている [1]．しかし，最もニーズが強く数も多いと考えられるドメインユーザ (第 2 章に詳述) の立場から見た場合，早速利用して自身の大規模計算を行える「簡単に構築できて効率的に実行できる」分散計算システムの方式が存在するとは言えない状況にある．つまり，ドメインユーザが手軽に利用だけ出来るシステムや簡単に分散プログラムが書ける方法が無い．

勿論，ドメインユーザが通常よく自身の計算に利用している OS や言語やライブラリではなく，分散システムの深い知識が要求される特別な OS や分散処理言語，ディレクティブやライブラリや関数の組み込み等を用いれば，分散計算が実現されることは既にずいぶん以前から充分知られている [1]．しかし，ドメインユーザにとっては，この様な複雑難解で異分野の OS や言語やライブラリの利用に基づく分散システムの知識は常日頃から修得済み，というものではない．ドメインユーザにとっては，必要に迫られる状況に立ち至ったときに早速に自身のプログラムに簡単に組み込んで利用だけ出来る，という条件でなければ，それは使えないシステムである．つまり，開発が面倒な分散システムは使われない．実際に短時間に計算が終わるからという理由だけのために面倒な開発を何ヶ月も行って利用しているというユーザは，現状では充分少ない．

現用の分散システムでドメインユーザに最も使われているのはMPIとPVMであるが，これとて，一時期は盛んに利用されてきたが[2]，今は，ハードウェアの急発展に押し流されてしまって，(少なくとも流体シミュレーションの世界では，研究用を除けば) 計算高速化の目的では余り使われていない．

そこで，学術としての分散方式や分散記述用言語の研究開発という観点ではなく，実用として分散計算がドメインユーザに手軽に広く利用されるようになるには，従来とは異なり「実用という観点」からのシステム構築とその実用性に関する評価が必要ではないかと考えた．これが本研究の動機である．

2 ユーザ要求と開発コスト

2.1 ドメインユーザの要求記述

本論文でいうドメインユーザとは，ソフトウェア開発やプログラミングの専門家ではなく，自身の専門分野を持ち，自家用プログラムを自身の仕事のために必要最低限開発するユーザである．現用言語としてはFortranが最も多く，新規な言語修得には消極的で，一連のソフトウェア開発プロセスやプログラミング技法等を意識しては適用しない．その専門分野としては例えば流体や構造の解析シミュレーション，画像分析，建築設計等である．プログラムの内容としては，特殊で試行錯誤の多い数百行から数千行プログラムを組む事が多い．

その様なドメインユーザは，分散処理方法自体等には関心は全くなく，計算が早く終わるという「結果」にのみ関心がある．これ以外のドメインユーザの要求は次のようになる．

- 現用の実行環境 (UNIX,Win98) と言語 (主としてFortran) を使いたい．
- 現用の逐次型のプログラムを出来る限りそのままで利用したい．
- プログラム開発・修正期間の短さと簡単さが最優先．(勿論,「無し」が最も良い)．
- システムに関する新たな学習は避けたい．

ただし，上記の要求を全て満たす分散型の低開発コストのシステムは困難なので，条件を一項目だけ緩め，言語をC++とし，オブジェクト指向記述を前提とした．C++はFortranに比べれば数値計算

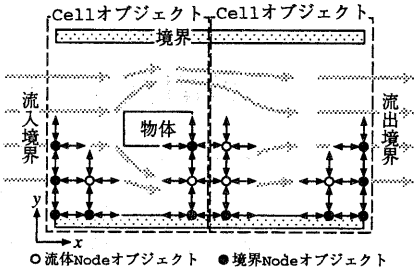

図 1: 数値風洞と計算領域分割の概念図

のユーザは少ないが，オブジェクト指向の持つモデル記述性の良さやクラスライブラリ再利用性の高さなどのため，複雑で大規模な数値計算においてドメインユーザにある程度利用されていると言って良い[3][4]．従って，ドメインユーザの心理的抵抗感や記述コスト高という点からしても最小限の条件緩和であると考える．

2.2 開発コストの定義

基本的考え方はプログラムの開発 (或いは改造) 開始から大規模計算 (複数回・多数回の試行や繰り返し計算を含む) の終了までの全過程を対象にしてそれに掛かる主要な「ユーザ負担」を総合的に評価する．本研究では既存の逐次型オブジェクト指向プログラムを分散型に改造する場合を対象とした．その場合，以下が項目として挙げられる．

- 改造・修正の方法の分かり易さ・煩雑さ
- 改造・修正に掛かる時間 (期間)
- 実行効率 (単一CPUとの処理速度比較)
- 新規な基礎知識習得に必要な時間

3 対象とした計算プログラム

分散システムに改造するための計算例の対象としたのは流れの世界で，空間二次元のナビエストークス方程式の連続流体流れを二次精度の陽解法の差分法の一種であるRoeスキームで解いた数値風洞のプログラムである．図1に示す様に正方形格子を切り，各格子 (メッシュ) の上下左右の物理量から自身の次の時点の物理量を計算する．大量のメッシュ点計算

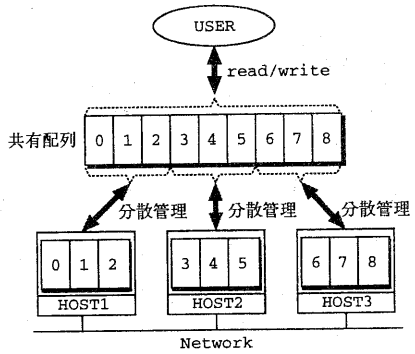

図 2: 共有配列の概念構成図

を必要とするため，計算領域を分割して (領域分割法) 分散計算にする．

4 分散化システムの提示と実装

まえがきで述べた課題を実現するため，その目標に合致した分散システムとして，以下の 2 種類の分散システムを提示する [5] [6]．また現用の代表的な分散化システムとして PVM ライブラリを利用したシステムを同じように構築し，比較・評価する．

4.1 分散共有配列を応用した分散化

4.1.1 分散共有配列 (DSA) の提案

DSA は，(共有配列と我々が呼んでいる) 分散共有メモリを提供する目的で開発された分散計算ライブラリである [5]．DSA ではメモリ上のアドレスを，計算機間で分散化されていることを意識しないで使える機能を実現しており，その機能を隠蔽し，ユーザには配列の形で使えるように提供し，自由に参照・変更が出来る機能を持たせた．DSA ライブラリでは配列の大きさや型，参照のタイミングを自由に決定することが可能である．同時に，DSA は同期制御のタイミングを自律的に設定する．ユーザは共有配列のある部分に目的データを記述し，他の計算機からアクセスすることで，分散環境内でのデータ転送 (交換) を実現できる．

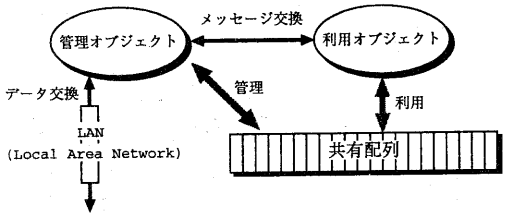

図 3: 管理オブジェクトと利用オブジェクト

4.1.2 DSA の構成

DSA の管理は，図 3に示すように管理オブジェクトにより配列内容の整合性を調べ，分散プロセスの同期制御，および共有配列の生成・削除等を行うことで実現された．ユーザが記述した分散化アプリケーションは利用オブジェクトと呼ばれ，と管理オブジェクトとのメッセージ交換で同期制御を実現し，管理オブジェクトは他の計算機の管理オブジェクトとのデータ交換を行うことで共有配列を実現する．

4.1.3 DSA の実現

DSA は分散処理の対象が大規模な数値計算であることから，そのアクセスと管理には高速性が求められる．また同時に大量データを扱うことも要求される．この条件を満たすため，バイナリファイルを作成することで大量データの扱いに対応させ，作成したバイナリファイルをメモリにマップすることで，高速なアクセスに対応させた．計算機間のデータ交換にも高速性が求められるので，スレッドを利用してデータ転送の並行処理を行うことで実現した．図 4に図 3を主要な構成要素の配置した DSA の全構成を示す．この図は HOST1,2,3 の 3 つの計算機で分散されている．

message stream は利用・管理オブジェクト間の通信に使用され，data stream は管理オブジェクト間の通信に使用されている．通信内容は計算の指示や共有配列データである．図 5に DSA ライブラリを使用した分散型ソースコードの雛型を示す．図 5には 5 種類の DSA ライブラリが挿入されており，それぞれ DSA のセットアップ，計算開始，計算終了，データ交換終了，DSA の終了を意味している．

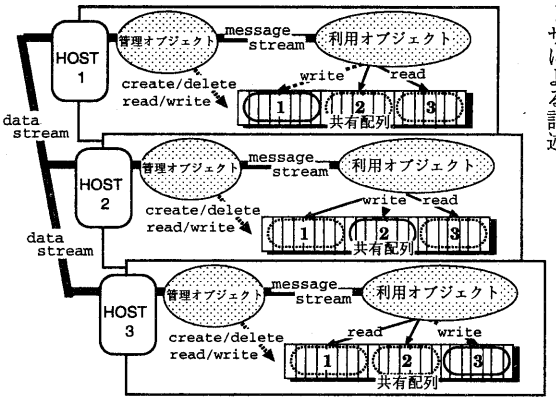

図 4: DSA システムの概念的な構成図

```
g_calculation.cc
#include "dsa.h"

main()
{
  g_c_setup(int, int);
    g_c_get_message();
    //計算部分
    g_c_send_wait_message(END_CALC);
    //データ交換部分
    g_c_send_message(END_CALC);
  g_c_quit(int);
}
```

図 5: DSA ライブラリの組み込み

4.2 自動分散化システム ADS

ADS は，逐次型のオブジェクト指向シミュレーションシステム (以降，OOSS と略記) から，計算負荷を均等化する自動分散型の OOSS を生成するシステムである [6, 7]．ADS では図 6に示すように

1. 通常の C++で記述された逐次型 OOSS のソースコードから，
2. 逐次型 OOSS でのクラスの Proxy オブジェクト・クラスのソースコードを生成して，逐次型 OOSS ソースコードに自動的に組み込むことで，
3. マスタ・スレイブ方式の自動分散型 OOSS のソースコードを，

生成する．なおスレイブ・プログラムのソースコードも ADS が同時に自動生成する．

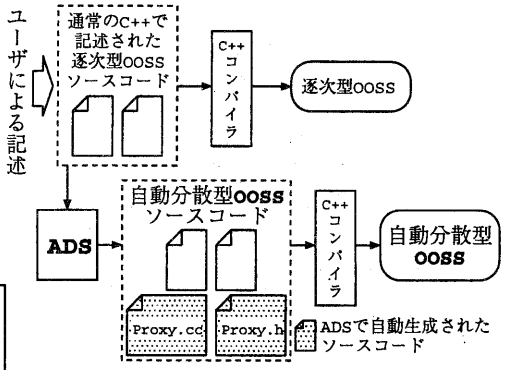

図 6: ADS を用いた自動分散化

この ADS で生成する Proxy オブジェクトは，隠蔽された分散通信処理，通信負荷軽減を考慮した分散オブジェクトの配置，非同期通信による並行処理の実現とその同期制御，デッドロック回避等の分散型 OOSS 実現に必要な機能を備えている．また生成された自動分散型 OOSS ではスレイブ配置に関する計算機負荷の均等化などの機能も含まれている．この Proxy オブジェクトの定義は全て ADS が行うので，ユーザが逐次型 OOSS を記述する際には分散処理や Proxy オブジェクトを意識する必要性そのものが無い．なお自動分散型 OOSS のコンパイルの手間は，逐次型 OOSS のそれと同程度である．

この ADS で生成された自動分散型 OOSS の実行性能は，スレーブ数 2 の分散計算で逐次型の約 1.7–1.9 倍程度の実行速度を実現している [6][7]．本研究の検証では通信負荷が高く分散計算に適さない計算法の OOSS を用いたが，そのような OOSS を自動分散化した場合でも実行効率が比較的良好であることが確認されている [7]．なお ADS ではソースコード解析処理の簡略化や分散計算を実現するために入力となる逐次型 OOSS の記述で若干の制限を加えているが，これらがユーザが逐次型 OOSS を記述する上で大きな障害にならない事も確認されている [6]．

4.3 PVM

PVM(Parallel Virtual Machine) は，ネットワークで接続された複数の計算機を単体のメッセージパッシング型並列計算機としてエミュレートすることを

表 1: 分散化システムの特性比較

	DSA	ADS	PVM
専用 API の提供の有無	有	少数	有
逐次型ソースコード記述の制限	無	有	無
ユーザが必要なシステムの理解度	中	低	高
ユーザによる同期制御記述の必要性	無	無	有

目的とし，現在最も主要なメッセージ通信ライブラリの一つである [1]．異機種間通信も考慮されているので，特に異機種間分散計算では多く用いらる．動的なプロセス管理機能や資源管理機能も持つ．

このPVMを用いたプログラミングでは，データの送受信関数や通信バッファの操作関数などを用いてプログラミングを行う．ユーザが分散処理を直接に記述するので分散計算を最適化できる半面，プログラミングするにはPVMのライブラリインタフェースを学習しなければならないので，どうしてもプログラミングを行うユーザに負担がかかる．またプログラム中へPVM関数が多数挿入されるために，ソースコードの可読性の低下が指摘される [1, p.183]．

5 開発コスト評価

5.1 分散化システムの特性評価

まず，各分散化システムから予め得られる特性を表1で比較する．上記の表から，最初の項目を除く3項目についてPVMがドメインユーザにとって「開発期間の長期化」を予測させる点を多く持っていることが分かる．最後の2項目「システムの理解度」，「同期制御記述の必要性」等はドメインユーザが特に避けたい項目であり，ドメインユーザからは低く評価される．DSAがADSに対して有利なのは，逐次型ソースコード記述の制限がない点である．これはドメインユーザにはメリットである．

5.2 実装作業結果の比較・評価

ライブラリ等の組込み作業の内容・項目を示す．

表 2: 分散コードへの記述変更個所数の比較

	DSA	ADS	PVM
システム稼働のための専用ライブラリ数	6	6	139
計算データ転送記述数	66	0	106
記述変更個所の合計	72	0	245

DSA の組み込みに必要な作業内容

- ヘッダファイルを追加
- 専用ライブラリを，ソースコードに追加
- 計算のループ構造を DSA の同期に対応させる変更を行う
- 共有配列へのアクセス記述を追加

ADS の組み込みに必要な作業内容

- ヘッダファイルを追加

場合によって以下の作業が必要．

- ADS が解釈できるよう，引数の型を変更･修正．

PVM の組み込みに必要な作業内容

- ヘッダファイルを追加
- 専用ライブラリを，マスター/スレーブの各ソースコード追加
- 専用ライブラリを用いた同期制御部を作成
- スレーブ間のデータ転送記述を追加

詳細に作業内容を検討すれば，容易にADSでは殆ど作業無しで済むことが多く，PVMではかなり複雑なシステム知識の必要な作業が多いことが分かる．DSAはその中間で，PVMに比べれば作業箇所数の比較でも少なく，内容的にも比較的簡単な知識で行える作業であることが理解できよう．表2にユーザのソースコードの記述変更箇所の比較を示す．この表も特にトータルではPVMの作業量が圧倒的に多いことを示している．

5.3 作業時間の比較

表3に実際の実装作業の結果のデータを示す．PVMの作業時間はPVM自体の移植時間が不定であり，最良ではDSAやADSと同等であるが，考察の項で述べるように作業者がシステムに多少詳しいものであったのでこういう結果になったと考えられる．本

表 3: 作業時間の比較

	DSA	ADS	PVM
システムに計算機を追加・移植に掛る時間	4 時間	2 時間	1 時間 ～3 日
分散コードへの記述変更作業時間	5 時間	1 時間	約 1 週間

来のドメインユーザであれば，もっと多くの差が出来たであろうと考えられる．この点を除けばPVMは全ての点で開発期間に他のシステムの何倍もの時間を要することが一目瞭然である．

6 考察

6.1 分散化システムの開発コスト

表 1，2，3 から総合的に分かるように，DSA は逐次型のソースコードを ADS のように自動で変換することは出来ないが，PVM ほど多くのライブラリをソースコードへ記述する必要もない．ユーザの実装作業量としては ADS と PVM の中間位置にあるが ADS に近い．DSA と ADS は PVM に比較すると開発コストにおいても，表 1，2，3 に見るように充分優位である．

従って，実行効率の高低によるドメインユーザのデメリットは殆ど差がないと仮定すると，分散システムのドメインユーザ評価は，開発期間とその煩雑さ如何によって決まると考えて良いであろう．であれば，我々の提案した，DSA，ADS の優位評価性は明らかとなる．本論文内には詳細な数値データを示せなかったが，定性的なデータと推察によっても上記の結論が否定される要素は殆どないといって良いと考える．(講演当日に，計測・評価データを示します.)

6.2 本研究の試行作業ユーザの考察

本研究では分散処理システムの概要とその実装経験があるユーザを被験者として実装を行った．従って，本研究の計算対象である Roe スキーム計算のソースコードの解析に 107 時間掛かった．ADS においては，この時間は殆ど必要がない．

Roe スキーム計算プログラムを作成した本人で，分散処理システムの概要と実装経験の無いドメインユーザであれば，ソースコードの解析時間はゼロ，システム実装法に対する (PVM よりは充分少ないけれどゼロではない) 多少の学習時間を掛けた後に，各システムの記述量 (個所) に比例した時間がかかると思われる．

7 結論と今後

本研究で提示した二方式の分散化システム DSA，ADS は「即座に使えるツール」という点では，他の分散化システムに比べて一日の長を持っていると評価できよう．今後は両システムの適用性の拡大や組み込みの簡潔さの高度化，より自動分散化されたシステムの構築を行う計画である．

参考文献

[1] 湯浅太一他編,「初めての並列プログラミング」, bit 別冊，共立出版，1998 年 6 月.

[2] Proceedings of the Parallel Computational Fluid Dynamics 95 Conference, Ecer A., Periaux J., Satofuka N., Taylor S.,(Eds.) held at Pasadena, Ca, USA, Elsevier, 1996.

[3] 日本機械学会：第 6 回計算力学講演会講演論文集 (特集：計算力学におけるオブジェクト指向アプローチ)，pp.35-67, 日本機械学会，1993 年.

[4] Atluri, S. N., Yagawa G., and Cruse T.A., (Eds.): "Computational Mechanics '95", pp.14-68, 1995.

[5] 上原　均，畠山正行,「UNIX プロセスのオブジェクト化」，オブジェクト指向'97 シンポジウム，in「オブジェクト指向最前線 (朝倉書店刊)」，pp.15 － 22，情報処理学会・ソフトウェア工学研究会，１９９６年 7 月 3 日．(共有配列)

[6] 上原均，畠山正行,「オブジェクト指向シミュレーションの自動分散化の実現と評価」，シミュレーション学会 (論文) 誌，第１７巻，第 4 号，pp.310-323, Dec., 1998.

[7] 上原　均，畠山正行,「自動分散型オブジェクト指向シミュレーションシステムの実行効率」，情報処理学会第 73 回 HPC 研究会，pp.19-24，98-HPC-73，1998 年 10 月 9 日.