数理モデル化と問題解決 11－2
（1997. 1. 24）

並列化支援視覚化システム NaraView におけるビュー間の連携方式について

笹倉万里子[1]　木和田智子[2]　城 和貴[2]　荒木啓二郎[3]
岡山大学工学部情報工学科[1]
奈良先端科学技術大学院大学[2]
九州大学大学院システム情報科学研究科[3]

自動並列化コンパイラは、逐次処理用に書かれたプログラムを並列実行用に自動的に書き換えることを目的としている。しかし、並列実行に関係するさまざまな条件をすべて考慮して最適な並列化を行なえるような評価方法は知られていない。そこでわれわれは、ユーザに、コンパイラがプログラムを解析して得た情報を視覚化して提示し、ユーザの指示によって並列化を行なうような並列化支援視覚化システム NaraView を提案する。本稿では、特に NaraView におけるビューと、そのビュー間の連携について述べる。

Views and their connection in NaraView

Mariko Sasakura[1], Satoko Kiwada[2], Kazuki Joe[2] and Keijiro Araki[3]
Okayama University[1]
Nara Institute of Science and Technology[2]
Graduate School of Information Science and Electrical Engineering
Kyushu University3

Parallelizing compilers aim to transform any sequential program to a parallel program automatically. But, general strategy of choosing the best transformation methods have not been known.Therefore, we are developing a visualization system named NaraView, which provides an interactive compilation or programming environment for parallel computer users. In this paper, we describe *views* and their connection in NaraView.

1 はじめに

逐次プログラムと並列プログラムがあり、かつ、同じ入力に対して両者の実行結果が同じである時、その逐次プログラムは並列化可能であるといい、逐次プログラムから並列プログラムへの書換えを並列化という。また、並列化を行なうコンパイラを並列化コンパイラと呼ぶ。

並列化コンパイラは、与えられた逐次プログラムを自動的に並列プログラムに書き換えることを目的としている。完全に自動的に書き換えるためには、実行時にしかわからない情報が必要であり、また、適切な並列化手法を選択するための戦略が明らかになっていなければならないが、現状ではこれらを並列化コンパイラに与えることができないため、完全に自動的に並列化を行なうことは困難である。そのため、ユーザがプログラムのどの部分をどの並列化手法を使うかを並列化コンパイラに明示的に指示して並列化を行なう。

ある逐次プログラムを並列化するためには、まずその逐次プログラムが並列化可能であるかを判定し、次に実際の並列化を行なうという2段階の処理を要する。ある逐次プログラムが並列化可能であるかどうかは、そのプログラムのコントロール依存関係とデータ依存関係によって判断できる。プログラムの並列化可能性の正確な判定には、コントロール依存関係とデータ依存関係の正確な解析が必要である。

しかし、ユーザがプログラムのソースコードだけを見て、コントロール依存関係や、データ依存関係を正確に把握するのは困難である。これらの依存関係は並列化コンパイラにより解析されるが、並列化コンパイラが解析した結果も、ユーザが見て簡単に理解できる形にはなっていない。

そこで、われわれは並列化コンパイラがプログラムを解析して得たコントロール依存関係やデータ依存関係を、ユーザが見て簡単に理解できるように視覚化を行なうシステム NaraView を構築している。NaraView は並列化コンパイラが解析した情報を視覚化し、ユーザの並列化作業を助けるという意味で、並列化支援システムであると言える。将来的には、NaraView からインタラクティブに並列化に関する指示を与えることができるように拡張する予定である。

これまでにも、並列化支援のためのさまざまな研究が行なわれてきている。VISTA[5] は、自動並列化コンパイラによって並列化されたプログラムの命令レベルでのフローグラフを図示し、ユーザがそれを参考にして、さらに並列に実行できる命令をインタラクティブに指示する環境を提供するシステムである。ParaScope Editor [2][4] は、ソースレベルでの並列化を支援するために依存情報を表示し、ユーザが依存を編集して実際に存在しない依存を取り除くことができる。また、どのような並列化手法を用いるかをユーザが選ぶことができる。しかし、情報をグラフィカルに視覚化する機能は備えていない。Paraassist[3] もソースレベルでの解析結果をもとに、ユーザとの対話を使って並列化を行うシステムである。しかし、ここでも情報をグラフィカルに表示する試みは行なわれていない。

NaraView では、ビュー という概念を用い、並列化コンパイラが解析して得たさまざまな情報を視覚化する。本稿では、NaraView で提供しているビューと、そのビューの間の連携について述べる。

2 NaraView の概要

NaraView はソースレベルでのコントロール・データ依存関係を視覚化することによってプログラムの並列化を支援するシステムである [7]。ここで、コントロール依存関係とは、条件分岐のように、文の実行順序を決めるプログラムの制御構造のことを意味し、データ依存関係とは、変数や配列 (今後これらをデータを呼ぶ) に対する書き込みアクセス、読み込みアクセス順番のように、文の実行順所を決めるデータ参照の関係のことをいう。

デバッグなどを目的とした他の多くのプログラム視覚化のアプローチでは実行結果から得られる情報を視覚化しているのに対し、われわれは、ソースプログラムだけから得られる情報を視覚化する。われわれの方法では、情報が比較的大雑把になり細かな最適化を行なえないと考えられるが、実行することなくある程度の情報が得られるならば、スケーラビリティやポータビリティの面からはその方が望ましい。

NaraView がプログラムを視覚化するために必要とする情報は、並列化コンパイラから得る。並列化コンパイラがソースプログラムの解析と並列化を行い、NaraView はそのコンパイラが行なった解析の結果を視覚化する。現在は、具体的な並列化コンパイラ

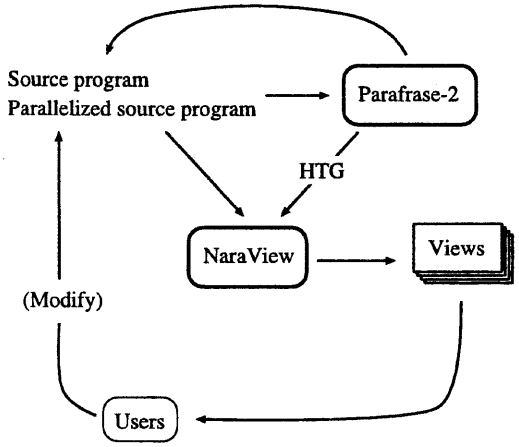

図 1: NaraView と Parafease-2

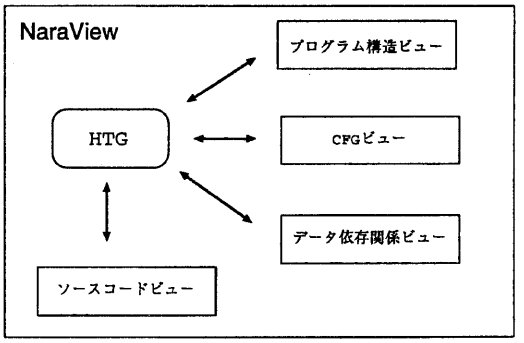

図 2: 各ビュー間の関連づけ

として、イリノイ大で開発された並列化コンパイラ Parafrase-2[6] を用いている。Parafrase-2 は Fortran および C で書かれたプログラムを並列化することができるが、本稿では Fortran のプログラムのみを扱っている。しかし、基本的な考え方はすべての手続き型言語で用いることができると考えている。

図 1に全体の構成を示す。ユーザは Fortran で書かれたソースコードを NaraView に入力する。すると、NaraView はまず、Parafrase-2 を呼び出し、並列化できる部分の自動並列化を図る。次に、NaraView は、ユーザによって入力されたソースコードと Parafrase-2 によって出力される解析結果を入力として、図を表示する。Parafrase-2 の解析結果は、コントロールフローグラフ (CFG) をループ構造よって階層化した HTG(Hierarchical Task Graph)[1] という形式で出力される。ユーザは表示された図を見ながらもとのソースコードあるいは Parafrase-2 が出力したソースコードを書き換える。これをプログラムが十分に並列化されたとユーザが判断するまで繰り返す。

3 ビュー

3.1 概要

並列化支援のためには、プログラムに関する情報をさまざまな観点から視覚化することが必要である。これを実現する一つの方法として、情報の入力形式と出力レイアウトを決定する枠組を複数用意することが考えられる。この方法を用いると、同じ情報を入力してもそれを異なる枠組で視覚化すれば異なる図が表示される。本論文ではこの枠組のことをビューと呼ぶ。

NaraView では次の 4 つのビューを提供している。

プログラム構造ビュー プログラム全体のプログラムの流れ、並列度、ループ構造、文の種類を 3 次元空間に表示する。

CFG ビュー ループの階層構造にしたがって各階層のコントロールフローグラフ (CFG) を 2 次元平面にグラフ表示する。

データ依存関係ビュー プログラム中のあるループについてそのループ内での変数に関するデータ依存関係を 3 次元空間に表示する。

ソースコードビュー プログラムのソースコードをテキストで表示する。

NaraView では、どのビューにおいても、HTG を通して図と視覚化対象のソースコードとの対応が明確になっている (図 2)。つまり、ビュー間の対応はすべて HTG で関連づけられる。これにより、例えば、プログラム構造ビューにおいて、あるループを指定すると、指定されたループの CFG ビューやデータ依存関係ビュー、ソースコードビューが表示されるようになっている。

ユーザはこれらのビューを使って次のように並列化を行なう。まず、プログラム構造ビューを使って並

列化を行なうループを決定する。そしてそのループのCFGビュー、データ依存関係ビュー、ソースコードビューを必要に応じて呼び出し、それらを参照しながら、直接ソースコードを変更して、実際には存在しないデータ依存を取り除いたり、並列化の方法やその適用順序を指定する。また、複数の並列化手法が適用可能な時は、それらを適用した結果の図を比べてより良い方法を選択する。以降で各ビューの詳細を説明する。

3.2 プログラム構造ビュー

プログラム構造ビューは、並列化を行なうループを決定できるように、プログラムを3次元的な「物体」として表示する。ここでは「物体」はノードの集まりとして実現される。一つ一つのノードはプログラムのソースコードの一文相当である。表示された「物体」に回転、拡大縮小を施すことにより、「物体」を様々な角度から眺めることが可能である。

各ノードは (x,y,z) の三つの値の組でその位置を表される。x, y, z のそれぞれにプログラムのどのような情報を対応させるかによって形成される「物体」の形状が変わってくる。ここでは、プログラムを直観的にとらえるため、また、並列化可能性をそこから読みとれるようにするために、次の情報を対応させる。

x: プログラムのながれ
y: 並列度
z: ループ構造に基づいた階層

プログラムのながれとは、プログラムの開始から終了までを表したもので時間軸と考えることもできる。並列度とは、一度にどれだけの文を実行できるかを示したもので、使用するプロセッサ数と考えることもできる。ただし、ここで扱うのはソースレベルなので、厳密な意味での使用するプロセッサ数ではない。階層とは、ループ構造に基づいてプログラムを分けたものである。これは現状では並列化が主にループについて行なわれることから、並列化可能な場所を特定しやすくするために用いる。

3.3 CFGビュー

CFGビューは、コントロールフローをグラフとして表示する。ソースプログラムの各文に一つノードを対応させ、その間をアークで結ぶことでコントロールフローを示す。

HTGはループ構造に基づいて階層化されたCFGなので、ここで表示されるのはそのうちの一つの階層のコントロールフローグラフとなる。ユーザはメニューなどによって任意の階層のCFGを表示することができる。

3.4 データ依存関係ビュー

データ依存関係ビューは、ユーザによる並列化手法の選択を支援する目的でデータ依存関係を視覚化する。データ依存関係では、次のことがわかるようにするのが重要である。

1. データ依存関係の有無。
2. データ依存関係がある場合、ソースプログラムを書き換えることによる並列化可能な部分の増加の可能性。

1のために、われわれは、まずデータのアクセスを表示し、その間をつなぐことによってデータ依存関係を表示する。その時、あわせて依存の距離を明確にするためにアクセス時間という概念を導入する。また、2を明確にするために、データアクセスとソースにおけるループの関係をループグリッドと呼ばれるものを使って明示的に表示する。

データ依存関係ビューでは、キューブ、ポール、ループグリッド、AVDマップを用いてデータ依存関係を視覚化する。

3.4.1 キューブ

キューブとは一つのデータアクセスに対し一つ表示される立方体である。一つのキューブの持つ座標値 (x, y, z) および色は次のようにデータアクセスの情報と対応する。

x,y　アクセスされるデータ
z　アクセスが行なわれる時間
色　アクセスの種類

データは x-y 平面上に配置される。配置の仕方はユーザが決めることができる。どの (x, y) 座標にどのデータが配置されているかは、後述するAVDマップで見ることができる。

データへのアクセスがいつ起こったかを示す単位として、ここではアクセス時間を導入する。アクセス時間はz軸に対応づけられる。データ依存関係の視覚化では、見たい情報によって、次の2種類のアクセス時間のうちどちらかを選ぶことができる。

ステートメントアクセス時間 文単位にアクセス時間を設定する。

イタレーションアクセス時間 イタレーション単位にアクセス時間を設定する。

ステートメントアクセス時間を用いれば、すべてのデータ依存関係を詳細に見ることができ、イタレーションアクセス時間を用いれば、ループキャリードのデータ依存関係を抽出してみることができる。アクセス時間はループ単位で設定することができるので、多重ループの場合外側のループをステートメントアクセス時間で、内側のループをイタレーションアクセス時間で表示することなども可能である。

アクセスの種類には、あるアクセス時間において読み込みのアクセスだけが行なわれる場合、書き込みのアクセスが行なわれる場合、両方のアクセスが行なわれる場合の3種類がある。それぞれ以下のように色と対応させる。

読み込みのみ	青
書き込みのみ	赤
読み込みと書き込み	紫

3.4.2 ポール

ポールとは、データアクセスを示すキューブとキューブをつなぎ、データ依存関係の存在を示す柱で、次の3種類がある。

ライフタイム・ポール 同一データに対して、先に書き込みアクセスがあり、その後読み込みアクセスが続けて一回以上起こっている場合、書き込みアクセスから、連続する読み込みアクセスの最後のアクセスまでをつないだものである。緑色で表示される。ライフタイム・ポールは、その始点となる書き込みアクセスで書き込まれた情報を保持しなければならない期間を示している。言い換えれば、始点となる書き込みアクセスで書き込まれた情報に対するすべてのフロー依存を示している。

アンチ-ディペンデンス・ポール 同一データに対して、読み込みアクセスが起こり、その次のアクセスが書き込みアクセスである場合、その読み込みアクセスから書き込みアクセスまでをつないだものである。黄色で表示される。アンチ-ディペンデンス・ポールは逆依存の存在を示している。

アウトプット-ディペンデンス・ポール 同一データに対して、連続して書き込みアクセスが起こる場合、その書き込みアクセスの間をつないだものである。ピンク色で表示される。アウトプット-ディペンデンス・ポールは出力依存の存在を示している。

これらのポールは、ポールの種類ごとに表示・非表示を選択することができる。

3.4.3 ループグリッド

データアクセスとループ構造の関係を明確にするために、指定したアクセス時間の位置にx-y平面に並行に半透明な平面を表示する。これをループグリッドと呼ぶ。ユーザは各イタレーションの最初、あるいは特定のイタレーションにループグリッドを表示するように指示することができる。また、ユーザは必要に応じてループグリッドの表示・非表示を指定することができる。

3.4.4 AVD(Array-Variable Disposition) マップ

AVDマップは、データがx-y平面上にどのように配置されているかを示す平面である。平面上のある点(x,y)に、そこに配置されているデータの変数名の頭文字を表示する。

一つの(x,y)座標に一つの変数、または配列の一つの要素を配置するのが原則であるが、いくつかの配列要素に対して一つの(x,y)座標を対応させることも可能である。前述したようにx-y平面上へのデータの配置の仕方はユーザが決めることができる。また、ユーザは必要に応じてAVDマップの表示・非表示を指定できる。

3.5 ソースコードビュー

他のビューで表示されている部分が、ソースプログラムのどの部分であるかを明確にするために、対応するソースプログラムを表示するのがソースコードビューである。各ビューでどの部分のソースコードを表示するかを指定する。その指定は、HTG の部分木を指定することと同じ意味を持つ。ソースコードビューは、指定された HTG 部分木に対応するソースコードだけを表示する。

4 例

ここでは、ガウスの消去法のプログラムを例として、NaraView のビューをどのように使って並列化を行なうかをを示す。ここで用いるガウスの消去法のプログラムは約 80 行の Fortran プログラムである。これを Parafrase-2 を用いて、自動的に並列化できる部分を並列化した後、ユーザの指示でさらなる並列化を行なう支援に NaraView を用いる。

図 3は Parafrase-2 で自動的に並列化した後のプログラムをプログラム構造ビューを用いて視覚化したものである。プログラムのながれは図の左奥から右前方向へ、並列度は左右方向、ループ階層は上から下に表示されている。これを見ると、左右方向に広がっているループが二つあることからすでに二つのループが Parafraes-2 によって自動的に並列化されたことがわかる。

どのループが並列化の可能性を持っているかの詳細をプログラム構造ビューから読みとることは困難である。しかし、どのループが並列化可能性が少なそうかを判断することはできる。例えば、条件分岐があるようなループでは、並列化できるかどうかを判断するのが難しい上、並列化した時の効果、すなわち、どれだけ多くの命令が並列実行可能となるか、を考えた時、一般的には多大なものを望めない。そこで、条件分岐が含まれないループに的を絞って並列化可能性を探るという方針を考えることができる。

プログラム構造ビューでは、条件分岐は黄色のノードで示される。図 3の矢印で示したループは、この条件分岐のノードを含まないループである。ループの並列化可能性を調べるため、これらのループを詳細に調べることにする。

これらのループのデータ依存関係を調べるために、データ依存関係ビューを起動する。データ依存関係ビューは、プログラム構造ビューにおいて、調べたいループを示すノードをクリックすることで起動される。ここでは、図 3で “Focused Loop” と書かれたノードをクリックしたものとする。その時表示されるデータ依存関係ビューが図 4 である。

図 4は、指定されたループをイタレーションアクセス時間で表示している。図では、アクセス時間は下から上へ示されている。指定されたループは 2 重ループであるが、そのうち、内側が並列化されていることが、プログラム構造ビューからわかる。ここでは、外側のループの並列化可能性を検討する。ループグリッドは外側のループに対応したものだけを表示している。

この図を見ると、並列化を妨げているのは、変数 m に対するアクセスであることがわかる。ソースコードビュー (図 5) を用いてソースコードを見ると、この m を配列化するスカラエクスパンションの手法を用いれば、このループは並列化できることがわかる。スカラエクスパンションを施した後のデータ依存関係を図 6に示す。確かに並列化できることがわかる。

5 おわりに

本稿では、並列化支援システム NaraView の持つビューと、ビュー間の連携について述べた。NaraView は Open-GL を用いた 3 次元グラフィックスで実現されている。本稿では図は白黒で掲載されているが、実際は色を使っているのでキューブやボールなどが見分けやすくなっている。NaraView のカラー出力は、
http://www.momo.it.okayama-u.ac.jp/~sasakura/NaraView.html
で見ることができるので、参考にされたい。

現在は、 NaraView を実用的に使える並列化支援システムとすべく拡張中である。並列化のためのユーザとのインタラクションや並列プログラムの実行時の通信状況の視覚化などを含めた統合的な並列化支援システムの構築を目標としている。

参考文献

[1] Girkar, M. and Polychronopoulos, C.: The Hier-

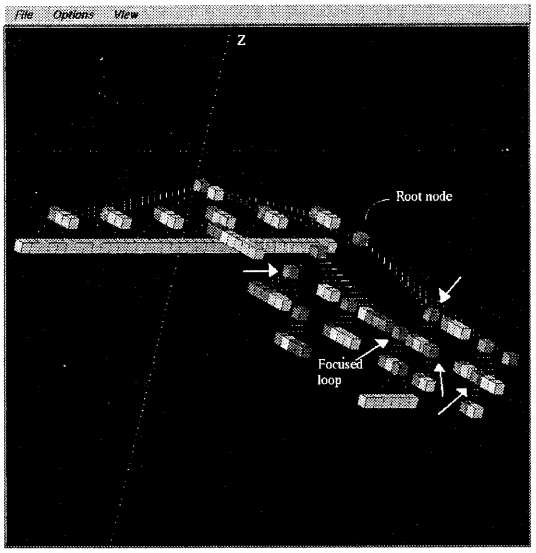

図 3: ガウスの消去法のプログラムのプログラム構造ビュー

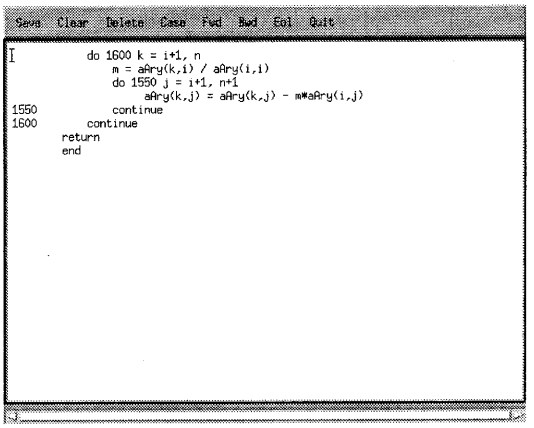

図 5: ソースコードビュー

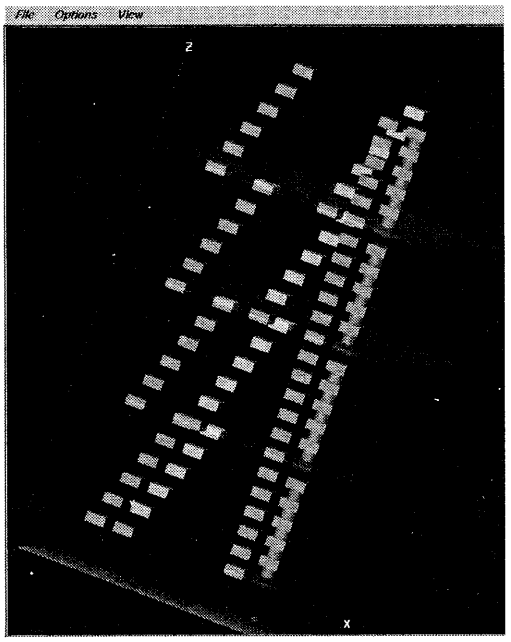

図 4: データ依存関係ビュー

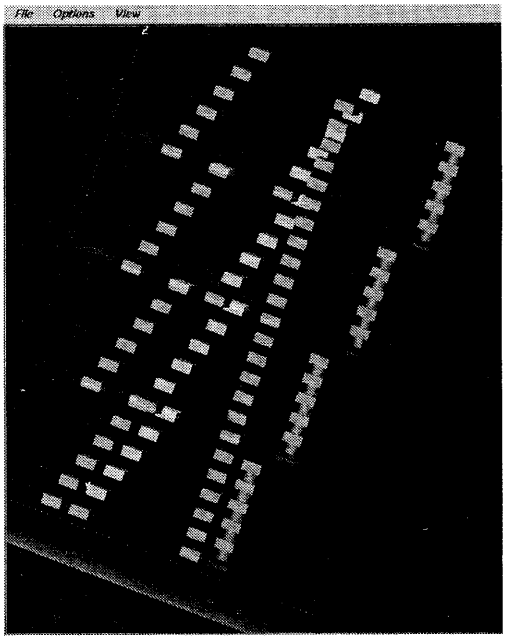

図 6: スカラエクスパンションを施した後のデータ依存関係

archical Task Graph as a Universal Intermediate Representation, *International Journal of Parallel Programming*, Vol. 22, No. 5, pp. 519–551 (1994).

[2] Hall, M., Harvey, T., Kennedy, K., McIntosh, N., McKinley, K., Oldham, J., Paleczny, M. and Roth, G.: Experiences Using the ParaScope Editor: an Interactive Parallel Programming Tool, *SIGPLAN Notice*, Vol. 28, No. 7, pp. 33–43 (1993).

[3] 岩澤京子, 黒澤隆, 菊池純男: 並列化支援システムによる Fortran DO ループの並列化方 法, 情報処理学会論文誌, Vol. 36, No. 8, pp. 1995–2006 (1995).

[4] Kennedy, K., McKinley, K. and C.-W.Tseng: Interactive Parallel Programming Using the ParaScope Editor, *IEEE Transactions on Parallel and Distributed systems*, Vol. 2, No. 3, pp. 329–341 (1991).

[5] Novack, S. and Nicolau, A.: VISTA: the visual interface for scheduling transformations and analysis, *Languages and Compilers for Parallel Computing. 6th International Workshop Proceedings*, pp. 449–60 (1993).

[6] Polychronopoulos, C. and et al.: Parafrase-2: An environment for parallelizing, partitioning, synchronizing, and scheduling programs on multiprocessors, *Proceedings of the 1989 International Conference on Parallel Processing* (1989).

[7] 笹倉万里子, 木和田智子, 城和貴, 荒木啓二郎: 並列化支援視覚化システム NaraView におけるプログラム 情報の 3 次元表示法, 並列処理シンポジウム JSPP'96, pp. 267–274 (1996).