並列プログラムの性能デバッギングを支援するアニメーション化ツール：かのこ

大澤範高[†,☆] 弓場敏嗣[†]

並列プログラムの性能デバッギング時の性能バグの発見には，可視化を用いることが有効である．ただし，並列プログラムの状態を表す大量のデータをそのまま可視化し，提示するだけでは人間の識別能力を有効に活用できない．人間が力学的なバランスや動作の変化に対して高い識別能力を有していることを利用するために，状態・統計量を座標や形状，色などのグラフィックス属性に直接マッピングするのではなく，馴染み深い力学系モデルへマッピングし，その系をシミュレーションした結果を3次元可視化・可聴化するアニメーションを提案してきている．並列プログラム実行時のトレースデータから提案のアニメーション化を行う性能デバッギング支援ツール「かのこ」を開発した．「かのこ」は，Javaプログラミング言語とVRML（Virtual Reality Modeling Language）を利用して実装されており，可搬性が高いツールである．本論文では，「かのこ」の機能と設計について説明し，実際のログデータと疑似ログデータのアニメーション化例のスナップショットを示す．さらに，提案のアニメーション化について考察する．「かのこ」は，並列プログラムの大域的性能デバッギング支援に役立つツールである．

An Animation Tool for Performance Debugging of Parallel Programs: Kanoko

NORITAKA OSAWA[†,☆] and TOSHITSUGU YUBA[†]

Visualization is useful to find performance bugs in performance debugging of parallel programs. However, simple visualization and presentation of a lot of data representing states of a parallel program do not utilize our human cognitive abilities to the fullest extent. The authors have proposed animation based on dynamic system modeling in order to better utilize the ability of human beings who can distinguish slight differences of dynamic system balance and changes of states. The proposed animation does not directly map state values and statistical values in program execution into graphical attributes such as coordinates, shapes, colors, etc., but it maps states in program execution into states in a dynamic system model, then simulates the dynamic system, and makes the result of simulation visible and audible in three-dimensions. A performance debugging tool *Kanoko* that uses the proposed animation has been developed. *Kanoko* is portable because it is implemented using the Java programming language and the VRML (Virtual Reality Modeling Language). This paper explains the functions and designs of *Kanoko*, and shows some snapshots of examples of animation. It also discusses the proposed animation. *Kanoko* is a useful tool to support global performance debugging of parallel programs.

1. はじめに

並列計算機やワークステーションクラスタが普及してきている．また，共有記憶型モデルに基づく並列計算に利用できるスレッドライブラリ仕様であるPthreads[9]やメッセージ交換ライブラリ仕様であるMPI[20]が標準的になっている．これらの環境の整備にともなって並列プログラミングが一般的になってきている．

しかし，効率の良い並列プログラムの開発は，逐次プログラムよりも難しい．不適切なプログラミングを行うと，並列性利用によって得られる計算の高速化よりも通信のオーバヘッドが大きくなり，十分な性能を得られない．並列プログラムの作成においては，適切な並列アルゴリズムを利用することだけではなく，データや計算の分散を環境に合わせて適切に行うことが必要である．並列計算機環境においては，論理的

† 電気通信大学大学院情報システム学研究科
Graduate School of Information Systems, The University of Electro-Communications
☆ 現在，文部省メディア教育開発センター
Presently with National Institutc of Multimedia Education

なバグの解消だけではなく，性能バランスの悪い部分（性能バグ）を見つけ，改善を行う性能チューニング（性能デバッギング）が重要である．クリティカルパスを短くすることがプログラムの高速化になる．高速化のためには，クリティカルパスを長くしている可能性のある，計算や通信のバランス不良箇所および計算と通信の間のバランスの悪いところを見つけることが性能デバッギングの第一歩となる．

並列プログラムの動作および性能の可視化の研究は行われている[8),16),19),21),27)]．従来の研究では，並列プログラムや並列計算機の状態を表す量もしくはその統計処理した値（状態・統計量）を，ある座標系における座標値，形状，色，明度などのグラフィックス属性に直接マッピングすることによって可視化が行われてきている．従来の性能デバッギングのための静的可視化の代表的なものとしては，2次元で表現するGantt Chart[6)]，Time-Space Diagram，Kiviat Figure[12)]がある．時間を含まない図（たとえば，Kiviat Figure）に時間軸を導入する3次元可視化も試みられている．また，時間の進行に応じて状態を変化させて表示するアニメーション化も行われている[19)]．しかし，可視化にあたって，馴染み深いものへの人間の高い識別能力を適切に活用しているものは少ない．数少ない馴染み深いものへの識別能力を活用した図の例には，人の顔形や表情を利用するChernoff's face[3)]がある．

我々は，状態・統計量をグラフィックス属性に直接マッピングするのではなく，馴染み深い力学系モデルへマッピングする並列プログラムの性能デバッギングのためのアニメーションを提案してきている[14),15),24),25)]．トレースデータから提案のアニメーション化を行うための可搬性が高いツール「かのこ」を開発した[26)]．その機能と設計について述べ，アニメーションによる並列プログラムの性能デバッギング支援について考察する．

2. 力学系モデルを利用したアニメーション

本章では，提案のアニメーション化について説明する．

可視化の目的は，人間の視覚的認識能力を十分に活用できるように，多数の値によって表される状態を適切に表現することである．したがって，大量のデータを単純にそのまま提示する方法では，可視化の目的を達成することはできない．人間が親しんでおり，判別がしやすい形式でデータを提示することが重要である．

人間は，力学的な世界で生活しており，力学系の動きになじんでいる．そこで，人間が力学的にバランスの良い状態とそうでない状態および力学的状態変化を容易に識別できることの活用を考える．状態・統計量を直接可視化するのではなく，一度力学系のモデルにマッピングし，そのモデルの動きをシミュレートし，シミュレーション結果を動的に可視化・可聴化する方法を提案してきている[14),15),24),25)]．この可視化によって，並列計算機環境での性能デバッギングにおいて重要な，通信の偏りや計算と通信のバランスの変化を馴染み深い力学系の動きとして表示することができる．アニメーション化は，**図1**に示す手順によって行う．この手順を以下で説明する．

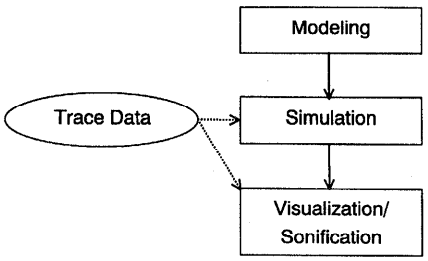


図1 アニメーション化手順
Fig. 1 Procedure of animation.

2.1 モデル化

まず，どのような力学的モデルを利用し，並列プログラムや並列計算機の構造をそのモデルにいかにマッピングするかを決める必要がある．並列計算機のネットワークトポロジもしくは並列プログラムにおける計算ノード間結合構造を基本としたモデル化ができる．たとえば，計算ノードを物体，計算負荷状態を物体の質量，相互結合網を物体間のバネ，通信を物体間の引力にマッピングすることができる．このマッピングでは，ノード間通信量が同一でもノード計算負荷が高いと質量が大きく，動きが小さくなる．これによって，並列計算環境（通信トポロジや計算能力）に負荷量や通信量が適合しているか否かを視覚的に確認できる．**表1**に，計算ノードを物体に対応させる場合に利用する力の例と効果を示す．

ここで，力学系へのマッピングは，厳密な物理的現実へのマッピングを必ずしも意味しない．物理的には起こりえない場合でも可視化において適切であれば利用する．たとえば，ノード間の通信量を物体間の力にマッピングする際には，物体間で1方向にのみ力が働くことも利用する．ノード間通信には方向があるからである．

次に，力学系モデルの動きをシミュレートできるように運動方程式を定式化する．系のエネルギー，散逸関数を求めてから運動方程式を生成するか，力の関係から運動方程式を直接生成する．

表1 計算ノードを物体に対応させる際に利用する力の例とその効果
Table 1 Examples of forces used in mapping a computing node to a body.

外力（引力）	通信などに応じて引力が働くことによって通信状況を物体間距離によって示す.
斥力	物体が集中してしまうことを防ぐ.
摩擦力	動きを滑らかにし，外力がなくなった場合に振動を止める．外力がある場合に系のエネルギーが増加しつづけることを防ぐ.
弾性力	通信を行うノード間（通信路）にバネのような力が働くことによってネットワークの構造を維持する.

2.2 シミュレーション

定式化した運動方程式（微分方程式の初期値問題）を数値的に解くことによって，動きをシミュレートする．動的な振舞いは運動方程式のパラメータによって変わる．このパラメータの変更によってユーザの着目する変化を強調できる．これは，カスタマイズ機能となる．

2.3 可視化

シミュレーション結果を従来の技法を利用して可視化する．従来の性能デバッギングのための技法だけではなく，シミュレーション結果の数値データを基にScientific Visualizationの技法を利用することもできる．

2.4 可聴化

従来の可聴化では，モノラル音の大きさや高さなどを変更して状態の変化を知らせる方法が一般的である[4),13)]．「かのこ」の可聴化では，音場の定位を利用した位置の特定を利用する．人は，音源との距離が大きい条件下で，音源と顔の中央線の間で約3°の角を識別できる[18)]．アニメーション内の物体から音を発生させることによって，位置やその変化をユーザに知らせることができる．また，音の利用によって，見えない箇所で着目すべき変化が起こったこと（起こること）を知らせることができる．

連続音を出し続けると耳障りなので，着目すべきデータの変化量（データの微分値）が一定の範囲になる際（範囲を超える際）に音を発生させることを基本とする．さらに，物体ごとに異なる音を指定できるようにする．

3. かのこ

これまでに，C言語，Perl言語[22)]，Virtual Reality Modeling Language Version 2.0（VRML 2.0）[1)]を利用したアニメーション化のプロトタイプツールを作成した[14),25)]．しかし，シミュレーション部はC言語，VRMLフォーマットへの変換はPerl言語，VRMLファイルの再生はブラウザによって行っており，シミュレーションパラメータや表示パラメータをインタラクティブに操作することはできなかった．

今回開発した「かのこ」では，図1におけるシミュレーションおよび可視化，可聴化をインタラクティブに行うことができる．VRML 97[10)]とJava言語[7)]を用いて実装しており，シミュレーションと可視化，可聴化を同一のブラウザ内で制御することができる．

「かのこ」の構造を図2に示す．VRMLによる記述部分とJavaで記述された部分，VRMLブラウザから構成される．VRMLのShape Nodeで物体形状の基本的な記述がなされ，Script NodeでJavaとのインタフェースがとられる．シミュレーション結果は，Javaプログラムから Script Nodeを介して座標変換をするための Transform Nodeに伝えられる．これによって物体が動く．物体の操作はVRMLのSensor Nodeによって感知され，Script Nodeを介してJavaで記述されたプログラムに伝えられる．別パネルでのノード情報表示やパラメータの設定などはJavaプログラムが処理を行う．視点移動などの3次元ナビゲーションは，VRMLブラウザの機能を利用する．

ユーザは，トレースデータファイルとして，トレースエントリのタイプ（ノード情報，リンク情報など），時刻，ノードIDもしくはリンクID，負荷統計情報，その他の情報を1行単位で記述したテキストファイルを用意する．負荷統計情報などからシミュレーションパラメータ（外力や質量）へのマッピングは，Javaで記述されたパネルを介して設定可能である．

VRMLもJava言語もインターネット環境においてよく利用されるようになっており，広い範囲のプラットフォームにおいて「かのこ」を利用することが可能である．また，トレースデータをWorld Wide Web（WWW）サーバに置くことによって，地理的に離れたユーザが協調して性能デバッギングすることも可能である．さらに，3次元空間のナビゲーションコマンドには人によって好みがあるが，適切なVRMLブラウザをユーザが選択できる．

「かのこ」の機能と設計について，シミュレーション，可視化・可聴化，インタラクティブ操作に分けてもう少し詳しく以下に説明する．その後，性能デバッギングにおける「かのこ」の基本的な利用法について説明する．

3.1 シミュレーション

運動方程式を解くJava言語で記述されたプログラムが，ユーザによって指定されたパラメータに従って

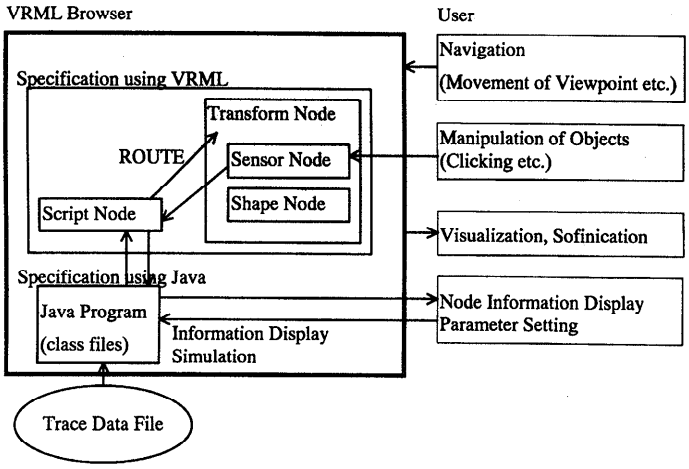


図2 「かのこ」の構造
Fig. 2 Structure of "Kanoko".

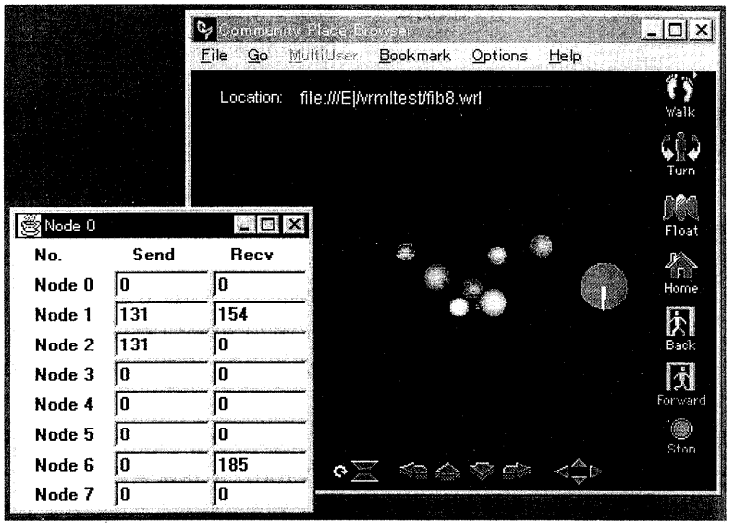


図3 「かのこ」の表示のスナップショット
Fig. 3 Snapshot of Display of "Kanoko".

シミュレーションを行う．物体間に働く力の強さのパラメータをインタラクティブに変更することが可能である．

モデルが複雑になった場合や物体数が多くなった場合には，シミュレーションに時間がかかる．また，同じトレースデータを可視化，可聴化する際にいつもシミュレーションを行う必要はない．そこで，シミュレーションと可視化，可聴化は別々に行っている．

3.2 可視化・可聴化

シミュレーション結果を基にして，Java 言語と VRML ブラウザの機能を利用して物体の動きや音を制御する．物体の識別のためのテクスチャや音をあらかじめ用意されている中から選択し，指定することができる．すなわち，可視化，可聴化のカスタマイズが可能である．小さいために分かりにくいが，**図3** では各球に異なるテクスチャを貼っている．

3.3 インタラクティブ操作

表示されている物体をクリックするなどして，アニメーションの進行を制御したり，性能デバッギングの対象の情報をインタラクティブに取得することができる．ノード情報を表示した例を図 3 に示す．図 3 の右側の針の 1 本ついた円盤は，シミュレーション時間の進行を表す時計である．12 時を示す場合が，シミュレーションの開始であり，時計回りに進行する．6 時

を示す場合が全処理時間の半分進行した時点を表す．円盤をクリックすることによって，アニメーションの再生を止め，再生速度などのパラメータを変更することが可能である．また，アニメーション停止時に物体（図3では球）をクリックすることによって，対応するノードの情報を得ることができる．再度円盤をクリックすることによってアニメーションを再開できる．

3.4 基本的利用法

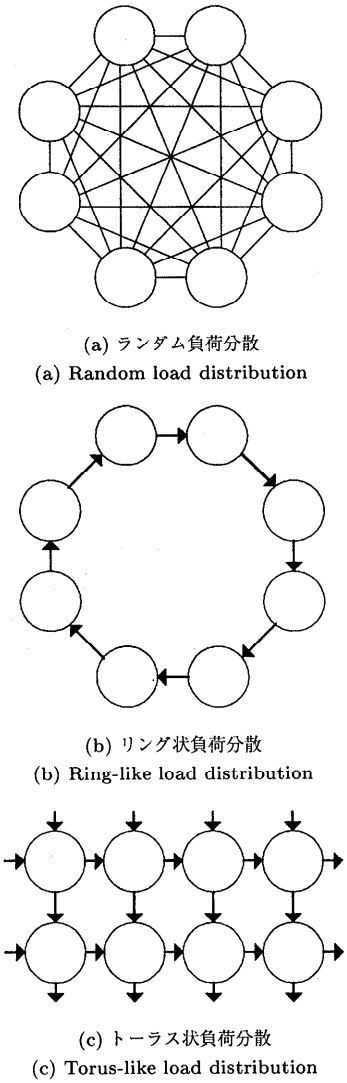
性能デバッギングの目的は，実行時間を短くすることであり，クリティカルパスを短くすることが必要である．クリティカルパスを長くしているのは，負荷分散が不十分なために計算時間がかかっている場合，通信に偏りがありボトルネックが生じている場合，不適切な負荷分散が行われているために通信時間がかかっている場合がまず考えられる．これらはそれぞれ，計算のバランスが悪い場合，通信のバランスの悪い場合，計算と通信のバランスが悪い場合と考えることができる．このような場合を見つけるために「かのこ」を次のように利用することができる．まず，対象並列プログラムを実行し，トレースデータを取得する．そのデータを「かのこ」に入力し，アニメーション化する．アニメーションを見ることによってバランスが悪いと思われる問題箇所の候補を探す．問題があると思われる時点でアニメーションを止め，その時点およびそれ以前のトレース情報，統計情報を検討する．また，必要に応じてソースコードを参照し，候補となった箇所が本当の問題か否かを調べる．問題であることが明らかになれば，その改善を試みる．改善した並列プログラムを実行することによって，実行時間が短縮されたかどうかを確認し，改善が不十分であれば，上記の性能デバッギングの過程を繰り返す．「かのこ」は，性能デバッギングの際の改善候補を見つけ，その状態を確認する作業を支援する．

4. アニメーション化例

例として，負荷分散方式による通信の偏りによる差異と，3次元メッシュ構造における通信と計算のバランスによる差異を示すアニメーションのスナップショットを示す☆．

4.1 負荷分散方式による通信の偏り

Fibonacci関数を並列計算機で実行させた際のトレースデータを基にしたアニメーションのスナップショットを示す．Fibonacci関数 $fib(n)$ の子関数 $fib(n-1)$ と $fib(n-2)$ をどのノードで実行させるかという負荷分散方式による違いを示す．



(a) ランダム負荷分散
(a) Random load distribution

(b) リング状負荷分散
(b) Ring-like load distribution

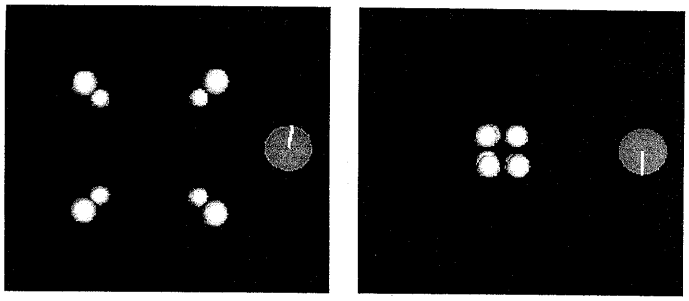
(c) トーラス状負荷分散
(c) Torus-like load distribution

図4 負荷分散方式
Fig. 4 Load distribution methods.

利用した負荷分散方式を図4に示す．子関数の実行ノードをランダムに選択する方法をランダム負荷分散と呼び，各ノードからの子関数実行ノードをただ1つとし，この関係がリングを構成する方法をリング状負荷分散と呼ぶ．また，子関数実行ノードの関係が2次元トーラス構造をとり，縦と横に1個ずつ送る方法をトーラス状負荷分散と呼ぶ．

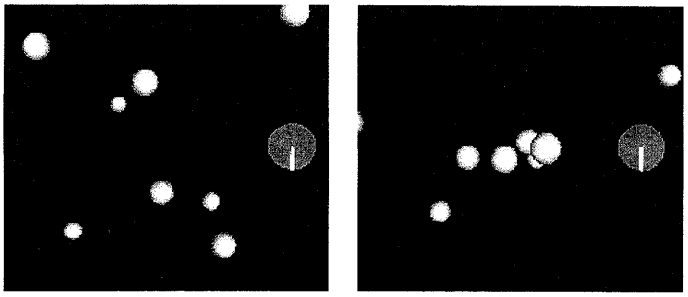
アニメーションに利用するトレースデータは，Fibonacci関数に引数22を与えて，要素プロセッサ数8の並列計算機Cenju-3[23]で実行させ，0.1秒ごとに

☆ アニメーション（VRML 97ファイル）は，次のURLからアクセス可能（1998年1月現在）．
http://www.yuba.is.uec.ac.jp/~osawa/anim/



(a) 初期状態
(a) Initial state

(b) ランダム負荷分散
(b) Random load distribution



(c) リング状負荷分散
(c) Ring-like load distribution

(d) トーラス状負荷分散
(d) Torus-like load distribution

図5 負荷分散状況スナップショット例
Fig. 5 Snapshots of load distribution states.

採取した．Cenju-3のノード間の通信レイテンシは均一である．この例でのトレースデータ採取間隔は粗いが，細かな間隔のデータを利用して可視化することも可能である．細粒度データを利用するとトレースデータが大きくなり，シミュレーションにかかる時間も長くなるが，アニメーション化に支障はない．

4.1.1 モデル化

ここでは，負荷分散方式による通信量の偏りを調べることを目的にモデル化を行う．ノードを物体に対応させ，斥力，引力，摩擦力を基にモデル化を行う．質量は一定とした．

まず，物体が離れて存在するように物体間に距離の自乗に反比例した斥力が働く．次に，送信量に応じて送信側にのみ受信側への引力（外力）が働く．送信側と受信側で非対称である．これは，作用反作用の法則に反するが，物体の動きによってノード間通信量の差が分かる．ノード間の相互通信が等量の場合は，重心に移動する．1方向が多い場合には偏りを生じる．最後に，速度に比例した摩擦力が働き，振動を減衰させる．これによって，外力による系のエネルギーの単調増加を防ぐ．

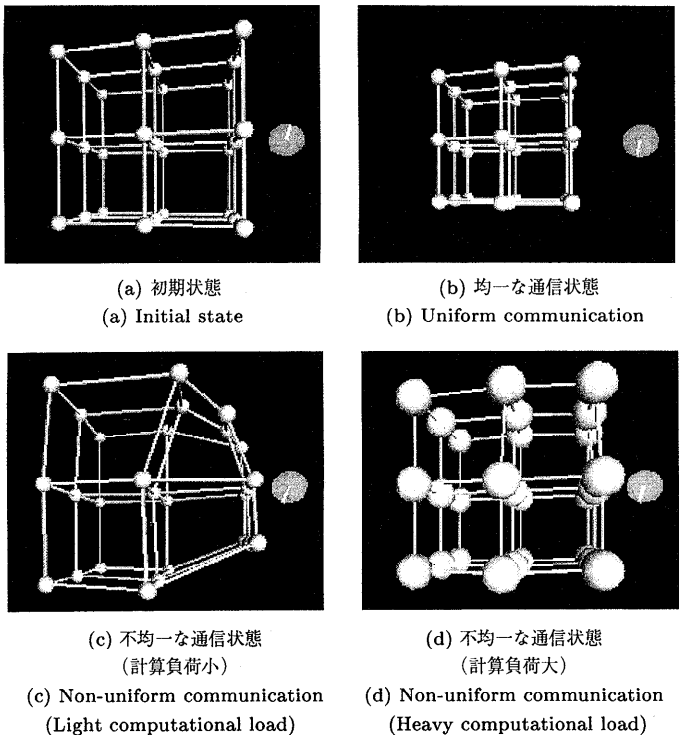
このモデル化による運動方程式は次に示す式で与えられる．右辺の各項が左から順に，斥力，引力，摩擦力を表す．$\mathbf{x}_i$ は物体 i の座標であり，$\mathbf{F}_{ij}(t)$ は時刻 t における物体 i から j に働く外力である．また，N は物体の総数であり，R，β は定数である．$|\mathbf{x}_i - \mathbf{x}_j|$ は，$\mathbf{x}_i$ と $\mathbf{x}_j$ との距離を表す．

$$\frac{d^2\mathbf{x}_i}{dt^2} = R\sum_{j=1,j\neq i}^{N}\frac{\mathbf{x}_i - \mathbf{x}_j}{|\mathbf{x}_i - \mathbf{x}_j|^3} + \sum_{j=1,j\neq i}^{N}\mathbf{F}_{ij}(t) - \beta\frac{d\mathbf{x}_i}{dt}$$

4.1.2 スナップショット

上記のモデル化とトレースデータに基づいたアニメーションのスナップショットを**図5**に示す．図5(a)が初期状態である．初期状態では立方体の頂点に球が配置されている．各ノードは球で表現され，球の大きさはすべて同一である．図では，遠近によって大小がみられる．計算ノードの負荷に応じて球の大きさや色を変えることが可能であるが，静的なスナップショットでは遠近感が曖昧になるのでここでは利用していない．各図の右側にある針を持った円盤の意味は，図3と同じである．

ランダム負荷分散の場合には，ほぼ均等に各ノード間で通信が行われ，物体間にほぼ均等に引力が働く．



(a) 初期状態
(a) Initial state

(b) 均一な通信状態
(b) Uniform communication

(c) 不均一な通信状態
(計算負荷小)
(c) Non-uniform communication
(Light computational load)

(d) 不均一な通信状態
(計算負荷大)
(d) Non-uniform communication
(Heavy computational load)

図6 3次元メッシュ構造のスナップショット例
Fig. 6 Snapshots of 3-D mesh structure.

これによって，図5(b)に示されるように物体が初期状態の重心の位置付近に集まる．

一方，リング状負荷分散の場合には，隣接ノード間にのみ力が働き，時間的にも通信量の偏りがある．このために図5(c)に示すように，リング状負荷分散では，均整のとれた形にはならない．ランダム負荷分散の場合とくらべて通信の分散が不適切であることを視覚的に容易に理解することができる．

トーラス状負荷分散の場合には，それらの中間的な状態になることが図5(d)から分かる．ある時点における表示を見ることによって，通信などの空間的偏りを知ることができる．一方，全体の大きさの変化を見ることによって，通信の時間的偏りを知ることができる．

4.2 通信と計算のバランス

図6(a)のように3×3×3の3次元メッシュ構造に配置されたノードとその間のリンクから構成される並列計算機において，並列プログラムを実行した際の通信の偏りおよび通信と計算のバランスの関係を調べることを目的としてモデル化を行う．

まず，隣接ノード間には構造を維持するための弾性的な力が働く．すべての物体間には距離の自乗に反比例した弱い斥力が働く．この斥力は，外力がなければほぼ初期形状に復元させるために利用する．この例では弾性力のみでは初期状態に戻らない．たとえば，弾性力のみの場合には，斜めに押しつぶした構造も安定である．通信によって物体間に外力（引力）が働く．引力は対称的である．また，計算ノードの負荷に応じて質量 $m_i(t)$ が変化する．これらのモデル化に基づいた運動方程式を下に示す．ここで L は，隣接物体間に弾性力が働かない距離である．C_i はノード i の隣接ノード集合であり，k は定数である．他の記号は，前節と同じである．

$$\frac{d^2\mathbf{x}_i}{dt^2}=\frac{1}{m_i(t)}\left[\sum_{j\in C_i}\left\{k\frac{\mathbf{x}_i-\mathbf{x}_j}{|\mathbf{x}_i-\mathbf{x}_j|}(L-|\mathbf{x}_i-\mathbf{x}_j|)+\mathbf{F}_{ij}(t)\right\}+R\sum_{j=1,j\neq i}^{N}\frac{\mathbf{x}_i-\mathbf{x}_j}{|\mathbf{x}_i-\mathbf{x}_j|^3}\right]-\beta\frac{d\mathbf{x}_i}{dt}$$

疑似的なトレースデータを基にした可視化の例を図6に示す．計算ノード，通信リンクが，それぞれ球と円柱によって表されている．球の体積は質量に比例させている．図6(a)が初期状態であり，メッシュ構造の交点に球が配置されている．均一な通信によって均

一な引力が働く場合には，図6(b)のように構造がほぼ保たれたまま縮小する．一方，不均一な通信が行われると構造が変形し，図6(c)に示される形になる．変形から通信に偏りがあることが分かる．また，図6(c)，(d)の通信量変化は同一であるが，計算ノード負荷による質量によって変形の程度が異なることが分かる．計算負荷（質量）が小さい図6(c)の方が，計算負荷が大きい図6(d)よりも変形の程度が大きい．このことから計算負荷と通信量のバランスを判断することが可能である．単なる通信の不均衡だけではなく，ノードの負荷との関係において通信量のバランスが不適切か否かを視覚的に判断することができる．

5. 考　　察

本論文のアニメーション化について考察する．

5.1 データの提示手法

Cenju-3のように各ノードがどのノードとも通信できる場合を考える．ノード間の通信量を，従来のようにリンク（通信路）ごとに折れ線グラフで表示することは可能である．しかし，ノード数の増加につれてリンク数が大きくなり，全体のバランスをみることが困難になる．たとえば，32ノードになると，有向リンク数は992となり，それらのグラフを一度にみて全体のバランスを理解することは困難である．

提案の方式を利用することによって，ノード数，リンク数が多くなっても大局的な通信の空間的，時間的偏りを容易に知ることができる．問題箇所を見つけた後に細部についての解析が必要であれば，従来の可視化手法を用いた図を用いることができる．提案の方式は従来の可視化手法を否定するものではなく，補完しあうものと考える．

5.2 アニメーション化の間接性

本論文の方式では，トレースデータをグラフィックス属性に直接マップせず，いったん，力学系にマッピングし，そのシミュレーション結果をグラフィックス属性にマップしている．つまり，間接的に可視化を行っている．力学系シミュレーションを用いることから初期値や働く力の履歴によって可視化された結果は異なる．

このために，アニメーションのある時点のスナップショットを見ただけで，その時点のノードの状態を逆に知ることはできない．しかし，アニメーション化による性能デバッグ支援は，スナップショットだけで性能バグを見つけられることを意図するものではない．静的なスナップショットではなく，不自然な動きなどから性能バグの可能性のある場所を探るきっかけにしようとするものである．

不安定な動きを示す箇所には性能バグの可能性があると考えられる．もちろん，不安定な動きがすべて性能バグとはいえない．一方，バランスが良い状態（動き）がある程度連続している場合には，基となったトレースデータもバランスが良いことがいえる．

また，履歴が影響することは必ずしも欠点ではない．バランスの小さなずれが時間とともに拡大して示されることもあるので，性能バグの発見に役立つ．

「かのこ」は，性能バグの可能性があると思われる箇所でアニメーションを止め，細部の状態情報を示す機能を有している．人間の力学的な動きやバランスに対する高い識別能力を活用して性能バグの可能性のある箇所を絞り，それぞれの箇所を細かく調べることができる．

5.3 可聴化の役割

現在の「かのこ」では，変化が大きな場合のみ音を発生させている．変化が激しい時期を音によって知ることができる．また，現在容易に利用できる2個のスピーカを利用するステレオ音によって音源（物体）の左右および奥行き方向の特定ができる．より多くのスピーカを利用することによって上下方向の特定も可能である．音が同時に複数箇所で発生しても音色，音階が異なれば個々の位置の判断ができる．さらに，音の発生時期がずれている場合には，そのずれを容易に認識することができる．時間的なバランスを知るためには可聴化が有益である．

また，現在のツールでは実現していないが，物体の動きの向きをより分かりやすくするためにドップラー効果を利用することも有益であろう．視点から見た物体の速度に応じて音のピッチを制御することによって実現できる．

5.4 ノード次数の大きな構造の可視化

各ノードがすべてのノードと通信できるような場合には，負荷分散方式による通信の偏りの例で示したように，各ノードをリンクでは拘束しない形でのモデル化および可視化が可能である．

また，高次のハイパキューブなどの構造は，3次元空間では，ノード間の距離などを自然な形で認識できるように配置できない．しかし，そのような構造においても，高次の構造を低次の空間に展開する方法を工夫することによって，より直感に反しない形で状態の変化を提示することが可能と考える．

5.5 大規模並列計算機への対応

現在の力学系シミュレーションの計算量は物体数をNとすると$O(N^2)$である．しかし，重力のように物体間の力が距離とともに急激に減少する力を利用する

場合には，適切な近似を利用することによって計算量を $O(N \log N)$ にすることができる[2]．単一プロセッサでは十分な性能が得られない場合には，シミュレーション部分の並列化が有効となろう．

また，物体数が非常に多くなった場合には，個々の物体を示す現在の可視化では判別が困難になる可能性がある．その際には，雲や煙として可視化を行う方法[17]が，データの傾向を示すために有効と考える．

5.6 パラメータの設定

アニメーションをより分かりやすくするためには，力学系モデルのシミュレーションをする際のパラメータの設定を適切に行う必要がある．パラメータの基本的な設定は，エネルギーの関係から設定することができる．

まず，平均的には外力によるエネルギー増加よりも摩擦力などによるエネルギーの散逸が大きい必要がある．この関係から外力の大きさと摩擦力などのエネルギー散逸のパラメータを設定することができる．

また，外力，摩擦力がない状態では，エネルギー保存則から構造に蓄えられるポテンシャルエネルギーと物体の運動エネルギーの和が一定である．物体の動きが速すぎたり，遅すぎたりして分からなくなることがないように平均速度を設定し，そこから平均運動エネルギーを求め，一定の変位で構造に蓄えられるエネルギーから弾性力や斥力のパラメータを設定することができる．より一般的にはリアプノフ関数によって系の安定性を議論することができる．

5.7 応　　用

提案の方式は，物体の接近・離散によって動作を示す．これを利用して性能データの可視化だけではなく，プログラムの実行状況を可視化することも可能である．さらに，摩擦力を大きくすると細部は見えなくなり，物体の偏りによって頻繁に実行されている部分を知ることができる．

また，本論文で示した力学系モデルへのマッピングとスナップショットは例にすぎない．ほかにも記憶領域を連続した弾性体（ゴムのようなもの）もしくはブロック（ページ）単位に分割された物体とし，記憶領域へのアクセスに応じて力が働くようにすることによってデータアクセスの状況をアニメーション化することが可能である．また，軌跡の併用によって識別をより助けることができる．

さらに，インターネットのノードをその地理的位置にあわせて，地形図とともに球上に配置し，球面は弾性的な性質を持つとし，ノード間のトラフィックに応じて力が働くようにすることで球の表面を変形させることができる．このマッピングによってトラフィックの変化を地形図の変化として見ることができる．

6. 関連研究

Spring model[11]や Force-directed placement[5]による静的な平面グラフ描画が研究されている．力学的なモデルを利用して描画する点は，本論文のアニメーション化と同じである．しかし，それらが安定解を求めることを基礎としているのに対し，「かのこ」は運動方程式を解くことを基礎としている点が異なる．また，アニメーション化の際にメッセージ送受信やノード負荷などのトレースデータに基づいて，外力や質量を動的に変化させる点も異なる．

運動方程式を解くのではなく，各時点で安定解を求め，それらを連続させることによるアニメーションでは，力学系モデルに従った自然な動きを表現できない．なぜなら安定解には時間の概念がないからである．たとえば，物体間に引力がある時間働くとすると，引力が働いているか否かによって安定解が大きく異なる可能性がある．単純にこれをアニメーションにすると動きが不連続となり，対応関係を見失う．また，複数の安定解が存在する場合には，初期値によって安定解が異なる．これもアニメーションにした際に不連続な動きを起こしうる．運動方程式を解く方法では解法に起因する不連続な動きは生じない．

ノードとリンクから構成される並列計算機構造を2次元もしくは3次元空間に配置し，計算負荷量や通信量に応じてリンクの太さや色などのグラフィックス属性を変化させることが可能であり，Pablo[21]などの既存の性能デバッガによって行われている．しかし，ノード数が多くなるとやはり分かりにくくなる．従来の性能可視化ツールの表示手法は，人間の力学的なバランスや動きに対する識別能力を活用するものではなく，「かのこ」とは異なる．

7. おわりに

本論文では，人間の識別能力を活用するために力学系モデルを利用した，並列プログラムの性能デバッギングのためのアニメーション化ツール「かのこ」について述べた．また，提案のアニメーション化の性能デバッギングに対する有効性について例を基に考察した．「かのこ」は，力学系モデルを利用したアニメーション化を行っており，大域的な性能バグの発見に役立つと考える．今後の課題は，より大規模な応用プログラムにおいて有効性を確認することと，性能デバッギングの目的に応じた標準的なマッピングを確立すること

である．また，モデル化をインタラクティブにできるようにすることとトレースツールとの統合は，使い勝手を向上させるために重要と考えている．

参考文献

1) Bell, G., Carey, R. and Marrin, C.: The Virtual Reality Modeling Language Specification Version 2.0, WD ISO/IEC 14772 (1996).
2) Bernes, J. and Hut, P.: A Hierarchical $O(N \log N)$ Force-calculation Algorithm, *Nature*, Vol.324, pp.446–449 (1986).
3) Chernoff, H.: The Use of Faces to Represent Points in k-Dimensional Space Graphically, *Journal of the American Statistical Association*, Vol.68, No.324, pp.361–368 (1973).
4) Francioni, J.M. and Jackson, J.A.: Breaking the Silence: Auralization of Parallel Program Behavior, *Journal of Parallel and Distributed Computing*, Vol.18, No.2, pp.181–194 (1993).
5) Fruchterman, T.M.J. and Reingold, E.M.: Graph Drawing by Force-directed Placement, *Software – Practice and Experience*, Vol.21, No. 11, pp.1129–1164 (1991).
6) Gantt, H.L.: Organizing for Work: Machine Record Charts, Progress Charts and Man Record Charts Show What to Do to Get Production, *Industrial Management*, Vol.58, No.2, pp.89–93 (1919).
7) Gosling, J., Joy, B. and Steele, G.: *The Java Language Specification*, Addison-Wesley (1996).
8) Heath, M.T., Malony, A.D. and Rover, D.T.: Parallel Performance Visualization: From Practice to Theory, *IEEE Parallel & Distributed Technology*, Vol.3, No.4, pp.44–60 (1995).
9) ISO/IEC: Information technology – Portable Operating System Interface (POSIX), Part I: System Application Program Interface (API) [C Language], ISO/IEC 9945-1:1996(e) (1996).
10) ISO/IEC: The Virtual Reality Modeling Language (VRML), ISO/IEC 14772-1:1997 (1997).
11) Kamada, T. and Kawai, S.: Algorithms for drawing general undirected graphs, *Information Processing Letters*, Vol.31, No.1, pp.7–15 (1989).
12) Kolence, K.W. and Kiviat, P.J.: Software Unit Profiles & Kiviat Figures, *Performance Evaluation Review*, Vol.2, No.3, pp.2–12 (1973).
13) Madhyastha, T.M. and Reed, D.A.: Data Sonification: Do You See What I Hear?, *IEEE Software*, Vol.12, No.2, pp.45–56 (1995).
14) Osawa, N., Morita, H. and Yuba, T.: Animation for Performance Debugging of Parallel Computing Systems, *Proc. 2nd Annual Symp. on the Virtual Reality Modeling Language (VRML97)*, pp.101–107 (1997).
15) Osawa, N. and Yuba, T.: Three Dimensional Animation for Performance Debugging Utilizing Human Cognitive Ability, *Proc. IFIP INTERACT97*, pp.102–103 (1997).
16) Reed, D.A., Shields, K.A., Scullin, W.H., Tavera, L.F. and Elford, C.L.: Virtual Reality and Parallel Systems Performance Analysis, *IEEE Computer*, Vol.28, No.11, pp.57–67 (1995).
17) Reeves, W.T.: Particle Systems: Techniques for modeling a class of fuzzy objects, *Computer Graphics (SIGGRAPH)*, Vol.17, No.3, pp.359–376 (1983).
18) Schmidt, R.F. (Ed.): *Fundamentals of Sensory Physiology*, Springer-Verlag (1986).
19) Simmons, M. and Koskela, R. (Eds.): *Performance Instrumentation and Visualization*, ACM Press (1990).
20) Snir, M., Otto, S.W., Huss-Lederman, S., Walker, D.W. and Dongarra, J.: *MPI: The Complete Reference*, MIT Press (1996).
21) Waheed, A., Kronmuller, B., Sinha, R. and Rover, D.T.: Toolkit for Advanced Performance Analysis, *Proc. IEEE International Workshop on Modeling, Analysis and Simulation of Computer and Telecommunications Systems*, pp.376–380 (1994).
22) Wall, L. and Schwartz, R.L.: *Programming perl*, O'Reilly & Associates (1991).
23) 広瀬，加納，丸山，中田，浅野，今村：並列コンピュータ Cenju-3 のアーキテクチャ，計算機アーキテクチャ研究会研究報告，SWoPP94, pp.121–128 (1994).
24) 大澤範高，弓場敏嗣：力学系へのマッピングによる並列計算機状態の動的可視化，第 53 回情報処理学会全国大会論文集，2E-03 (1996).
25) 大澤範高，弓場敏嗣：力学系モデルを利用した並列計算機動作状態のアニメーション，第 4 回インタラクティブシステムとソフトウェアに関するワークショップ（WISS'96），pp.189–197 (1996).
26) 大澤範高，弓場敏嗣：並列プログラムの性能デバッギングを支援するアニメーション化ツール「かのこ」，並列処理シンポジウム（JSPP97），pp.305–312 (1997).
27) 長谷川隆三，越村三幸：並列プログラムおよび性能デバッギングのための視覚化，コンピュータソフトウェア，Vol.12, No.4, pp.33–44 (1995).

(平成 10 年 1 月 26 日受付)
(平成 10 年 9 月 7 日採録)

大澤 範高（正会員）

昭和58年東京大学理学部情報科学科卒業．昭和60年同大学大学院理学系研究科情報科学専攻修士課程修了．昭和63年同博士課程修了．理学博士．ソフトウェア開発会社を経て，平成5年電気通信大学大学院情報システム学研究科助手．平成10年文部省メディア教育開発センター助教授．並列分散システムソフトウェアに興味を持つ．ACM，IEEE-CS各会員．

弓場 敏嗣（正会員）

昭和16年9月22日生まれ．昭和41年3月神戸大学大学院工学研究科修士課程修了．野村総合研究所を経て，昭和42年通商産業省工業技術院電気試験所（現，電子技術総合研究所）に入所．以来，計算機のオペレーティングシステム，見出し探索アルゴリズム，データベースマシン，データ駆動型並列計算機などの研究に従事．その間，計算機方式研究室長，情報アーキテクチャ部長などを歴任．平成5年4月より，電気通信大学大学院情報システム学研究科教授．並列処理一般に興味を持つ．工学博士．電子情報通信学会，日本ロボット学会，日本ソフトウェア科学会，ACM，IEEE-CS各会員．