時系列データの統計解析によるPCクラスタシステム解析手法の提案

山村周史[†] 平井聡[†] 小野美由紀[†]
松本和宏[†] 住元真司[†] 久門耕一[†]

本論文では，PC クラスタシステムにおける新たなシステム解析手法を提案する．本手法の特長は，時系列データを入力とし，統計解析の 1 手法であるクラスタ分析を時間方向とノード方向の両方に対して行う点にある．本手法により，PC クラスタシステム上で採取した大量の測定データ（関数プロファイル情報および CPI 等の CPU アーキテクチャ性能情報）から性能ボトルネックとなりうるシステムの挙動変化が，いつ・どのノードで発生したかを迅速に検出し，その原因を特定できる．我々は，本手法を実装した統合解析ツール *iScopes*（Integrated Software for Comprehensive Performance Evaluation System）を開発し，8 ノード（16CPU）で構成される PC クラスタシステムへの実適用を行った．その結果，異常ノードの発見やアプリケーション分析に対する本手法の有用性を確認できた．

System Analysis Method by Statistical Analysis on Time-series Data for a PC Cluster System

Shuji Yamamura,[†] Akira Hirai,[†] Miyuki Ono,[†]
Kazuhiro Matsumoto,[†] Shinji Sumimoto[†] and Kouichi Kumon[†]

In this paper, we propose a novel system analysis method for a PC cluster system. The feature of our method is to use cluster analysis (one of the statistical analysis methods) on time-series data for both directions of time and multiple nodes. Our method can quickly identify the performance bottleneck which can impact the behavior of the PC cluster system from a large amount of measurement data. Our target measurement data includes functional profiling data and CPU performance information from the architectural viewpoint. We developed *iScopes* (Integrated Software for Comprehensive Performance Evaluation System) implementing our method. Our evaluation result on the 8 nodes (16 CPUs) PC cluster system shows its effectiveness to identify the irregular node and analyze a parallel application behavior.

1. はじめに

近年，主に HPC 分野において PC クラスタシステム（以下,「PC クラスタ」と略記）の利用がさかんであり，その規模は数ノードで構成される小規模なものから数百ノードを超える大規模なものまで幅広い．現在，HPC 分野で利用されている PC クラスタの構成は大規模化する一方であり，その計算ノード数は急速に増加する傾向にある．

クラスタ規模が大きくなると，調査対象となるシステム構成要素が増加するため，システム障害発生時において障害ノードを切り分けたり，性能問題発生時にその原因を発見したりすることが困難となる．実際には，原因を特定するために各々のノードを逐一分析する必要が生じ，人的・時間的なコストの増大を招く．特に，ある特定のノードの性能劣化が時系列で極希にしか発生しないような場合，その調査は困難を極める．

また，HPC アプリケーションの効率的なチューニングを行う場合には，各ノード上で時系列での処理の違い（初期処理，計算処理，通信処理，終了処理等）を把握して，各処理ごとにチューニングを施すことが有効である．具体的には，処理ごとの実行時間や CPI（Clocks Per Instruction ）・キャッシュミス率等の CPU アーキテクチャ性能情報（以下,「CPU 性能情報」と略記）を収集して活用できることが望ましい．

そこで我々は，HPC アプリケーション動作中のシステム全体の挙動を簡単かつ迅速に把握したいと考えた．これを実現するためには，再コンパイルやソースコード修正等をいっさい行わず，アプリケーションや OS，デバイスドライバを含めたシステム全体の挙動を，時間方向とノード方向（時間的・空間的）の両方

† 株式会社富士通研究所
FUJITSU LABORATORIES LTD.

向に対して把握することが必要となる．しかしながら，この要求を満たすためには，性能情報をすべてのノードで時系列で連続的に計測して，分析しなければならない．この場合，ノード数に比例してデータ量が増大し，分析者の負担がきわめて大きくなるという課題がある．

本論文において，我々は上記の課題を解決し，HPC アプリケーションが動作する PC クラスタの迅速な挙動把握を行うことを可能とする新たな解析手法を提案する．本手法は，時系列データを入力として，時間方向とノード方向の両方向に対して統計処理を適用することによって大量のデータに対する効率的な分析を実現する．本手法を実装した統合解析ツール「iScopes」を開発し，PC クラスタへの適用を行い，実システム上でその有用性を確認した．

以下，2 章において，PC クラスタにおけるシステム解析の課題を述べる．次に 3 章において時系列データを対象とした統計処理によるシステム解析手法を提案する．続いて，4 章において本論文で提案する iScopes の基本構成とその実装について詳述し，5 章で本ツールによる手法の評価を行う．6 章において関連研究について述べ，最後に 7 章で本論文を総括する．

2. PC クラスタにおけるシステム解析の課題

我々は，これまで，小規模なクラスタから RSCC[1] 等の大規模 PC クラスタの開発や性能分析，チューニング，運用等を行ってきた．本章では，PC クラスタ上で発生する障害原因の発見やアプリケーション分析を行ううえで，実際に直面してきた課題について述べる．

2.1 時系列での詳細なシステム把握

PC クラスタ上で HPC アプリケーション稼働中に，時系列で障害や性能劣化が発生することがある．実際に発生した事例として，ユーザが意図しないデーモンプロセス（kjournald 等）が定期的に起動し，その周期が偶然にもシステムのロードアベレージ算出の周期と一致したことがあった．この場合，CPU 負荷率を誤って出力するという問題が発生した．ほかにも，デーモンプロセスの起動やネットワーク性能の動的な変動により，一時的な性能のばらつきが発生するという問題も発生した．また，CPU のメモリバス飽和によってノード内の CPU 数を増加させても性能向上が得られないといった問題が発生したケースもある．

このような現象を解析するためには，時系列で採取された詳細なシステム情報が必要となる．その情報として，ソフトウェア上は CPU 使用率内訳（USER：SYSTEM：IDLE）や OS，デバイスドライバ，各プロセスの関数別実行時間内訳（プロファイルデータ）があげられる．また，ハードウェア上は，CPI やキャッシュミス率等があげられる．これらの情報を統合して分析できることが望ましい．

しかしながら，既存の関数プロファイラ（gprof[2]，OProflie[3]）や MPI ログ機能の組合せでは，分析に有用な時系列データを得ることは難しい．デーモンの起動やアプリケーション内部の挙動を把握するためには，時系列データの精度が少なくとも数秒単位でなくてはならない．しかも，HPC アプリケーションの実行時間は通常，数十分以上動作することが多い．既存の関数プロファイラでは，上記精度で測定の開始と終了を繰り返し，そのたびにディスクへの I/O が発生するため測定オーバヘッドが大きくなるとともにデータサイズも増大する．さらに，gprof のような単一アプリケーションの調査を目的としたツールは，OS やデバイスドライバ，他プロセスを含めた関数プロファイルを出力できない．

2.2 対象システムを構成するプログラムへの変更を必要としないデータ採取

実際の性能測定の現場では，システムの変更（システム内のプログラムの入れ換えや再コンパイル）が困難な場合が多い．2.1 節で述べた詳細情報を測定し，システム全体の挙動を把握するために，システム上のすべてのプログラムを入れ換えたり，再コンパイルを行ったりするためには多くの手間がかかり，短時間での計測を要求される現場では対応が難しい．さらに，OS やデバイスドライバを含んだプログラムの変更によって，システムそのものの動作が変化して本来の障害解析が行えないという危険もある．

したがって，対象システムを構成する OS やプログラムへの変更をいっさい必要とせずに，2.1 節の時系列の詳細情報をどのように採取するかが課題となる．既存のツールはプログラムの変更を前提としており，その適用が難しい．たとえば，PerfExplorer[4] は，時間方向に対するアプリケーションの挙動分析を行っているが，動作内容が既知のアプリケーションに対してソースコード修正もしくは動的パッチによるプローブポイントの挿入で対応している．我々が目的としているプログラムに対する変更なしにシステムを解析するためには十分とはいえない．

2.3 膨大な測定データの効率的な分析

システム挙動を把握するために，すべてのノードについて 2.1 節で述べた時系列の詳細データを採取した場合，計算ノードの台数増加に比例してデータ量が膨大となる．たとえば，5 秒単位で関数プロファイルを

作成した場合，わずか 10 分程度の測定でも 120 個のプロファイル結果を表示する必要が生じる．このとき，システム全体では（120 × ノード台数）個のプロファイル結果を画面に表示しなければならず，解析は事実上不可能となる．通常，HPC アプリケーションの実行時間は，数十分以上であることが多く，さらに解析が困難となる．以上から，どのようにして大量の時系列データを表示し，それを解析して有益な情報を見出すかが課題となる．

3. 時系列データの統計処理によるシステム解析手法の提案

本章では，2 章で述べた課題を解決する新たなシステム解析手法について述べる．

3.1 時系列データによる詳細なシステム挙動の解析

任意の時間間隔ごとに OS，デバイスドライバおよびシステム上で動作する全アプリケーションの関数プロファイルおよび CPU 性能情報を採取する．これにより，時系列で情報が得られるようになり，すべての計算ノード上で本情報を採取することで，「いつ」，「どのノードで」，システムの挙動がどのように変化したかが分かる．同時に，特定のアプリケーションに着目すれば，その動作内容が時間方向に対してどのように変化するかを関数レベルで分析することも可能となる．

実際に，PC クラスタ上で姫野ベンチマーク[5] を実行中，特定ノードにおいてシステム状態監視コマンド「sar」を起動し，時系列データを測定した結果を**図 1** および **図 2** に示す．このグラフは，採取した関数プロファイルおよび CPU 性能情報から 1 秒ごとにプロセス単位の CPU 使用率（棒グラフ）および CPI（折れ線）を表示したものである．これらの図から明確に分かるように，約 10 秒間隔で sar コマンドが実行され（図 1 中の斜線部），これと同期して他ノードを含めて CPI が同時に大きく変化している．このように，時系列でデータを観測することで，プログラム実行中のいずれの時間帯にどのようにシステム挙動が変化したか，またどのようなプログラムや関数が動作したかを容易に確認することができる．

3.2 プログラムの修正を必要としないデータ採取

プログラムの修正を行わずに関数プロファイルおよび CPU 性能情報を測定する．そのために，アプリケーションや OS（カーネル）とは独立して計測処理を行う．

関数プロファイルについては，ハードウェア割込みによるサンプリング手法により測定を行う．gprof や PerfExplorer のように，コンパイル時に計測用関数の呼び出しルーチンを挿入したり，メモリ上にロードされたプログラムへの動的パッチによって実行回数を計測したりする等の直接的なプログラム修正は避ける．割込みハンドラは，割込み発生時に走行中のプロセス ID や命令アドレスを記録し，測定後に記録されたプログラムの外部シンボルをマッピングする．

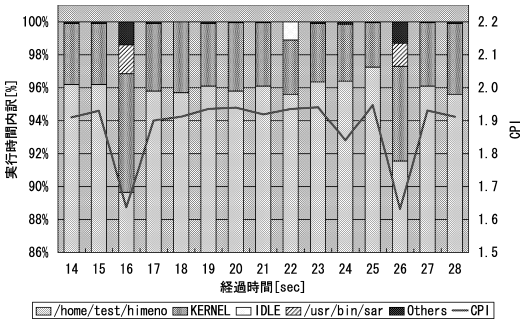


図 1 時系列データ例（sar コマンド起動ノード）
Fig. 1 Example of time-series profiling method (the node with invoking sar command).

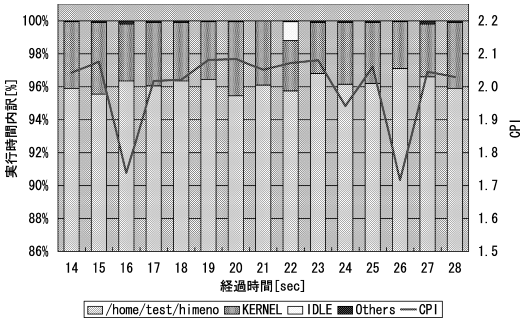


図 2 時系列データ例（sar を起動していないノード）
Fig. 2 Example of time-series profiling method (the node without invoking sar command).

CPU 性能情報については，性能モニタリングカウンタ（Performance Monitoring Counter：以下「PMC」と略記）[6] を用いることで計測できる．ただし，PAPI[7] 等の既存の PMC ツールのようにソースコード内部に直接 API を埋め込むのではなく，プログラム外部で PMC を制御する．この制御は，前述の関数プロファイリングを行う割込みハンドラと合わせてカーネルモジュール内に実装すればよい．PMC を停止・再開を一定間隔で繰り返すことにより，時系列の CPU 性能情報を生成することができる．

3.3 統計処理による大量データの処理

多数台のノード上で測定された大量の時系列データ処理に対して，「類似したデータどうしを少数のグループにまとめる」という統計処理手法を導入する．

一般に，PC クラスタ上で動作する HPC アプリケーションは，時間方向とノード方向で動作内容を考えた

場合，いくつかのグループに類別できると期待できる．たとえば，領域分割による並列化を行った場合には，データの境界部分を担当するノードと中間部分を担当するノードでは処理内容（負荷）が異なる．このような動作内容のグループ化は，平均値以上/以下や標準偏差（分散）を用いた 2 値的なデータ分析では十分とはいえない．そこで，我々は，データを統計的にグループ化することによってこれを実現する．

すべてのノードで測定した大量の時系列データを入力として統計処理を行うことにより，下記のような効果が期待できる．

- 時間方向に関して，たとえば，初期処理，本計算処理，終了処理のように，計算処理の過程により挙動をグループ化してデータの傾向を説明できること．
- ノード方向に関してグループ化することで，異常発見や各ノードが担当する処理内容を説明できること．

以上のように，時系列データを採取し，統計解析によりデータをグループ化することで，PC クラスタに対し，時間方向およびノード方向の両面で挙動を正確かつ容易に把握することができるようになる．

4. iScopes の設計とその実装

我々は，前章で述べた解析手法を統合解析ツール iScopes として設計および実装を行った．本章では，これについて述べる．

4.1 設　　計

統合解析ツール iScopes は，以下の点を考慮して設計を行った．

(1) HPC アプリケーションを対象アプリケーションとすること．また，単一ユーザが占有して利用する環境（シングルユーザ環境）を主な測定環境とすること．
(2) 対象 OS として Linux，対象 CPU として複数の IA（Intel Architecture）プロセッサをサポートすること．
(3) 大規模クラスタ環境にも対応可能な低オーバヘッドな測定が行えること．
(4) OS やアプリケーションへのソースコードの修正や再コンパイルを必要としないこと．
(5) HPC 用途の PC クラスタにおいて，時系列データをグループ化するのに最も適した統計解析手法を選択すること．

(1) については，iScopes の目的は HPC アプリケーションの性能評価であり，多くの場合，ベンチマーク実行はシングルユーザ環境で行われる．したがって，iScopes では，主としてシングルユーザ環境を主な測定環境とする．

(2) および (3) を実現するために PMC を用いた．PMC は，ほぼすべての IA プロセッサに搭載されているハードウェア機構であるため，異種の IA プロセッサに同機構により対応できる．PMC へのアクセスは，カーネルモード（特権モード）命令が必要なため，カーネルモジュールによりその制御を行う．PMC はハードウェアで実装されているため，計測に際してのオーバヘッドも小さく抑えられる．

特に (3) については，大規模クラスタにおいても測定の開始やデータ回収処理が性能に悪影響を与えないように考慮する．測定の開始・終了指示は，現行の 1000 ノードを超えるような大規模環境で利用実績がある単純なコマンド（rsh 等）のみを使用する．また，データ回収時のディスクやネットワークへの負荷増大の影響を避けるため，測定区間内では回収処理は行わず，当該区間後（あるいはアプリケーション実行終了後）に独立して行う．

(4) については，gprof で必要となる「-pg」等の特殊オプションを付加する等の追加作業を行わずに計測できることを考える．そのために，OS やデバイスドライバ，システム上の全ユーザプロセスを含むシステム全体に対して PMC オーバフローによるサンプリング手法を適用するとともにエクスポートされた関数シンボル情報を組み合わせることで対応する．

(5) の統計解析手法の選択については，以下のような検討を行った．

一般に，統計解析において大量のデータを入力とし，類似するデータどうしを少数のグループにまとめる手法として，階層型クラスタ分析，非階層型クラスタ分析，自己組織化マップ等が知られている．

このうち，非階層型クラスタ分析および自己組織化マップは，階層型クラスタ分析とは異なり，グループ化の初期配置をツールのユーザが設定する必要がある．また，解析結果が乱数に依存する場合があり結果の評価が難しい等，統計解析の非専門家がツールを利用するという観点から使い勝手が悪いと考えられる．

一方，階層型クラスタ分析[8),9)] に関して，最長距離法，最短距離法，Ward 法等が利用可能である．このうち，最長距離法には，分けられた個々のグループが小さくまとめられる特質がある．いいかえると，個々のグループに属するデータが互いによく似ているということである．このため，個々のグループごとに実際の測定値との対応がとりやすい．計算機性能のデータ

表 1 iScopes の動作環境
Table 1 Machine environment for iScopes.

測定環境	OS	Linux 2.4.x / 2.6.x
	ディストリビューション	Red Hat 8/9, EL 2.1/3.0/4.0, Fedora Core 3
	CPU	Pentium 3, Pentium 4, Xeon Itanium 2, AMD64, Opteron
分析環境	OS	Windows 2000 / XP

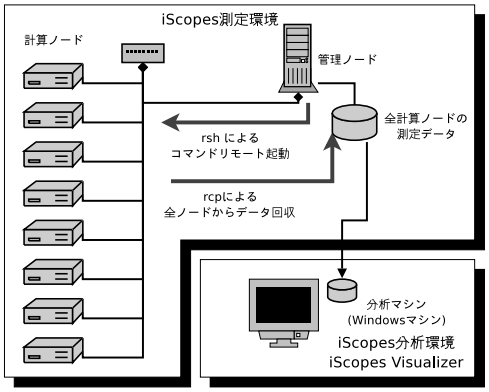


図 3 iScopes のシステム構成
Fig. 3 iScopes system overview.

解析の観点からは，最長距離法が最も良いと考えた．

以上から，iScopes の統計解析手法として最長距離法を評価基準とする階層型クラスタ分析（以下，単に「クラスタ分析」と略記）を採用した．

4.2 構成と実装

iScopes の動作環境を**表 1** に，全体構成を**図 3** に示す．この図は，1 台の管理ノードおよび 8 台の計算ノードで構成された PC クラスタの例である．iScopes は，大きく「測定環境」および「分析環境」の 2 つの環境で構成される．

測定を開始する際，管理ノード上で測定コマンド `pcc_is_run` を起動する．このコマンドは，基本的には，内部で（1）rsh（SCore[10] 環境下では rsh-all）コマンドにより各計算ノード上で時系列データ採取を開始，(2）測定完了後，rcp コマンドにより測定データの回収，という動作を行う．測定途中では，システムへの影響を避けるため，ディスクやネットワークへのいっさいの I/O 処理を行わない．測定データはメインメモリ上の十分小さいバッファ領域に一時的に保持し，測定区間終了後にディスクへの書き出しと管理ノードへの回収処理を行う．分析を行うユーザは，管理ノードに回収された測定データを Windows マシン上で動作する iScopes Visualizer で読み込み，GUI による対話的な解析作業を行う．

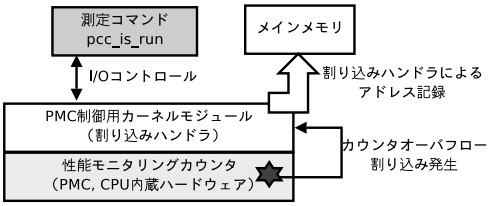


図 4 iScopes のソフトウェア構造
Fig. 4 iScopes software structure.

測定に際して，iScopes が必要とする機能は rsh による `pcc_is_run` コマンドの起動と測定終了後の rcp による回収のみである．rsh/rcp は，大規模 PC クラスタ上でも利用実績があり，測定中のディスクやネットワークへの I/O 処理を行わないため，PC クラスタの大規模化によるアプリケーション実行への影響はほとんど発生しない．また，ディスク使用量については，1 ノードあたりに記録用として確保するメモリサイズ × ノード台数が管理ノードに確保できればよい．後述するように，通常測定時のメモリサイズは 1 分あたり約 1 MB である．この場合，たとえば 1,024 ノードで 30 分間測定しても 30 GB の領域が確保できれば十分であり，大規模なクラスタでもディスク容量は大きな問題とはならない．

iScopes で測定する時系列データは，(1）時系列で採取した関数別実行時間内訳（「時系列関数プロファイリングデータ」と呼ぶ）と（2）時系列で採取した CPU 性能情報（「時系列 CPU 性能情報」と呼ぶ）の 2 種類である．これらは，いずれも PMC を使用して計測を行う．

（1）時系列関数プロファイリングデータ測定部の実装

図 4 に PMC およびソフトウェア構造を示す．図のように，ソフトウェア部は PMC 制御用カーネルモジュール（ドライバ）と測定コマンドで構成される．PMC 制御モジュールは，OS の修正や再起動の必要がなく，測定時に動的にロードして利用可能である．このモジュールは，プロファイリングを行うための CPU 内部イベント[6] およびサンプリング周期を設定する．PMC のカウンタオーバフロー時に割込みが発生し，本モジュール内の割込みハンドラ内でアプリケーションの PID（4 バイト）と命令アドレス（4 バイト）をメインメモリ上に記録する．メインメモリは，CPU ごとに PMC 制御用カーネルモジュールが確保する．メモリサイズは，システムの搭載メモリ量に依存するが，2CPU 時に 1 ms サンプリングで 1 分あたり約 1 MB（(4 B+4 B)× 60,000 × 2 = 960 kB）で記録可能で

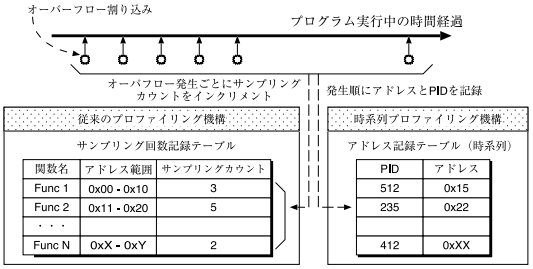


図 5 従来のプロファイリングの基本機構と時系列プロファイリング機構の違い
Fig. 5 Difference between a conventional profiling mothod and our time-series profiling method.

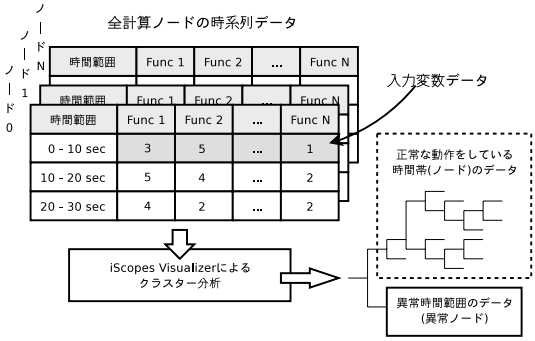


図 6 iScopes Visualizer によるクラスタ分析
Fig. 6 Cluster analysis with iScopes Visualizer.

ある．

PMC を用いたサンプリング手法そのものは既存の OProfile[3] 等と同様である．しかしながら，割込みハンドラ内でのデータ記録方法に大きな違いがある．**図 5** にその違いを示す．従来の PMC を用いるプロファイラは，対象プログラムの関数シンボル，アドレス範囲およびサンプリング回数記録エントリで構成されたテーブルをあらかじめ作成する．そして，割込み発生ごとにサンプリング回数をインクリメントする手法を採る．一方，iScopes は，割込みハンドラ内部で PID および命令アドレスのみを時系列で連続的に記録する．関数シンボルとのマッピングは測定後に行うことで，低オーバヘッドで時系列のプロファイルデータを採取できる．本記録方式により，測定期間内の任意の時間範囲のプロファイル情報を自由に生成することが可能となる．なお，測定の後処理として必要となる関数シンボルとのマッピングに要する時間は，測定データのサイズによるが，8 ノードで約 30 分間測定を行った場合のデータに対して，3GHz 程度の Pentium 4 を用いて数分で完了する．

iScopes は，割込みハンドラ処理を極力単純化することで，1ms 間隔程度のサンプリング周期での測定オーバヘッドは約 1%以内に抑えている．上記の時系列データを，従来の PMC を用いるプロファイラで測定する場合には，任意の時間間隔で測定の開始・中断，データ収集処理を行う必要があり，大きなオーバヘッドをともなう．そのため，実際には同様の時系列プロファイリングデータを採取することは困難である．

（2）時系列 CPU 性能情報測定部の実装

図 4 と同様の機構を用いる．ただし，オーバフロー割込みは使用せず，一定時間ごとに PMC 値の記録とリセットを行うという単純な構造である．

（3）クラスタ分析の実装

iScopes によるクラスタ分析を**図 6** に示す．入力変数には，時系列プロファイリングデータを用いる．各時間帯の関数別実行時間を 1 データ要素として各関数の実行時間を入力する．

クラスタ分析を行う iScopes の問題点は，何グループに分類するか（グループ数）をユーザが決定しなければならない点にある．元来，階層型クラスタ分析は大量のデータを分析するためのユーザ補助手法であり，不用意に自動化すると誤った分析結果を導き出す危険もある．そのため，我々は GUI による対話的なアプローチが最も適していると考え，iScopes Visualizer による GUI 実装を行った．実際に，データを分類する場合には，クラスタ分析結果を適宜参照しながら最も適切なグループ数をユーザが試行錯誤を繰り返して発見しなければならない．

また，前述のように，サンプリングデータそのものは CPU ごとにメモリ上に記録しているが，クラスタ分析時には，ノード単位で CPU ごとのデータをまとめたものを入力とする．したがって，ノード内部の CPU 間で挙動が大きく異なる場合には，詳細分析を行うことはできない．しかしながら，iScopes が対象とする HPC アプリケーションでは，ほとんどの場合ノード内部（SMP 構成）での計算・通信処理の内容が同じになるため，処理速度や分析画面上での簡略表示等を優先させ，ノード単位で一括処理する．

5. iScopes の評価

本章では，時系列データを入力としたクラスタ分析による解析手法を実装した iScopes により，本手法の有用性を評価する．

5.1 測 定 環 境

実験に使用した PC クラスタの構成（計算ノード）を**表 2** に示す．本 PC クラスタは，計算ノードと同スペックの管理ノード 1 台および計算ノード 8 台で構成される．各計算ノードに 2CPU ずつ搭載されてお

表 2 PC クラスタの構成（計算ノード）
Table 2 Specification of the PC cluster system (calculation nodes).

計算ノード数（CPU 数）	8 ノード（16 CPU）	
クラスタシステムソフトウェア	SCore Cluster System Software バージョン 5.8.3	
計算ノードの構成	CPU	Intel Xeon 3.60 GHz × 2
	メインメモリ	2 GB
	NIC	Intel 82546 GB Gigabit Ethernet Controller （rev 3）
	OS（カーネルバージョン）	Linux（2.6.11-1SCOREsmp）
	ディストリビューション	Fedora Core release 3

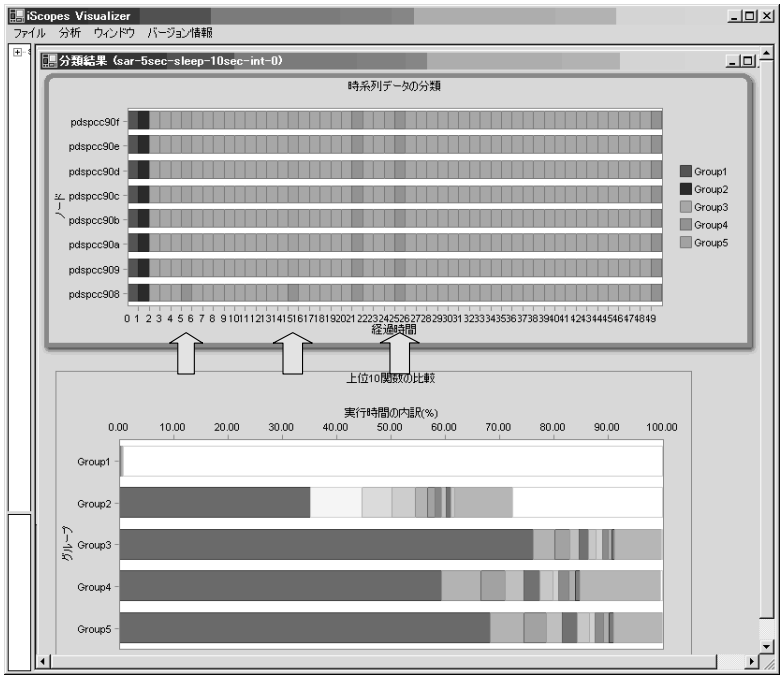


図 7 姫野ベンチマーク実行中の異常を発見した iScopes Visualizer のスナップショット
Fig. 7 Snapshot of iScopes Visualizer detecting the irregular state during the execution of Himeno Benchmark.

り，測定時，各ノードあたり計算プロセスを 2 個ずつ起動する．本クラスタを管理するシステムソフトウェアは，SCore version 5.8.3 を使用した．

5.2 異常動作ノード発見とその原因特定

表 2 の PC クラスタ上で姫野ベンチマークを実行し，iScopes によるデータ採取・分析を行った．姫野ベンチマークのデータサイズは Medium を選択した．実行中，異常模擬を行うために，1 台のノード上のみで sar コマンドを 10 秒ごとに 3 回起動する．iScopes による時系列データ測定結果は 3.1 節で説明した図 1 および図 2 のようになった．

ここで，iScopes Visualizer による分析時のスナップショットを**図 7** に示す．この図では，姫野ベンチマーク開始から 50 秒後までの測定結果を 5 つにグループ化して色分け表示している．グループ数は，最初，ある程度多いグループ数（例．32 程度）を選択し，図下部に表示されるプロファイル結果を参照しながら，同様の動作を行う項目がまとまるように徐々にグループ数を減少させて決定した．

この図から分かるように，sar コマンドが起動された図中最下部のノードでは，CPI の変動が発生した時間帯がクラスタ分析によって自動的にグループ分けされていることが分かる（図中の矢印部分）．また，画面表示については，ここでは分かりやすくするために全ノードを表示しているが，上部 7 ノードについてはまったく同じ動作を行っており，画面上では 1 ラインとして簡略表示も可能となる．このように，計算ノード数が増加してデータ量が増えた場合でもコンパクトに解析結果を可視化できる．

分類されたグループに基づいた姫野ベンチマークの詳細なプロファリング結果を**表 3** に示す．この表は，異常発生（sar 起動）の 1 秒間について関数別実行時

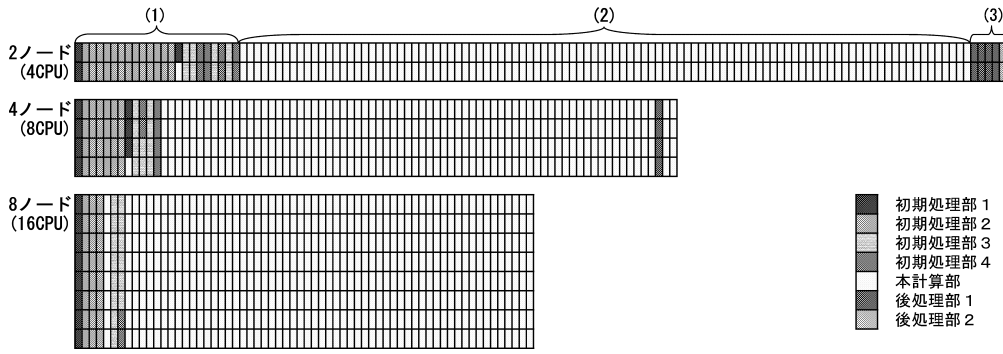


図 8 PHASE の時間方向での処理内容の時間変化をクラスタ分析した結果
Fig. 8 The result of cluster analysis on time-series behavior with iScopes Visualizer (PHASE).

表 3 正常動作時と異常動作時のプロファイリング結果の違い
Table 3 Difference of profiling results between regular state and irregular state.

	処理内容	G-3	G-4	G-5
			異常時	
		正常時 [ms]	sar 起動ノード [ms]	非 sar 起動ノード [ms]
姫野ベンチ	主計算処理	745	616	632
	受信処理	103	147	174
	MPI 関数	78	94	111
	その他	34	42	49
OS		39	71	33
sar		0	15	0
その他		1	15	1

間の内訳を示したものである．表中，G-3，G-4，G-5 は，クラスタ分析によりグループ化された番号 3〜5 を表している．

この表から分かるように，まず，sar 起動ノード（図 7 内の最下部ノード）「G-4」では，正常時「G-3」と比較して受信処理や OS，sar の実行時間が増加している．これは，sar コマンドの起動により該当ノードの姫野ベンチマークの主計算処理が滞り，他ノードがデータ受信待ち状態になったためである．図 1 と合わせて分析すると，G-4 と同時間帯（実行開始から 16 秒後および 26 秒後周辺）において，CPI が 1.9 から 1.7 まで減少している．これは，主に受信待ちループ処理が増加したことによるものである．他方，非 sar 起動ノード（図 7 内の上部 7 ノード）「G-5」では，正常時「G-3」と比較して sar および OS の実行時間に変化はないが，受信処理は増加していることが分かる．図 2 と合わせて分析すると sar 起動ノードと同様に当該区間において CPI が減少している．これも，sar 起動ノードと同様に受信待ちループ処理の増加によるものである．

以上のように，iScopes はアプリケーション実行中のどの時間帯にどのようなシステム挙動の変化が発生したかを迅速に検出できる．クラスタ規模が大きくなった場合においても，姫野ベンチマークに代表されるような定常的な動作を行う HPC アプリケーションの場合，同様の計算処理を行う時間帯およびノードが増加するのみであり，クラスタ分析による異常の検出や画面内へのコンパクトな可視化が実現できる．加えて，当該範囲の関数レベル/CPU 性能情報レベルでの解析も可能である．

表 4 PC クラスタ上での PHASE の実行時間
Table 4 Elapsed time of PHASE on the PC cluster system.

計算ノード数	実行時間 [sec]	2 ノードに対する性能比
2	657.2	1.0
4	413.3	1.6
8	312.2	2.2

5.3 並列アプリケーションの並列化効率の解析

文部科学省 IT プログラム戦略基盤ソフトウェアの 1 つである，第 1 原理擬ポテンシャルバンド計算ソフトウェア PHASE[11] を実行し，iScopes によるデータ採取・分析を行った．PHASE は，ナノプロセス解析をはじめとして幅広く利用されている MPI アプリケーションである．テスト用の比較的小さな入力データを使用し，計算ノード数を 2，4，8 と変化させた場合の実行時間を**表 4** に示す．この表から分かるように，計算ノードを増加してもスケールしていない．

iScopes により分析を行った結果を**図 8** に示す．この図は，5 秒間隔で測定した時系列データに対してクラスタ分析を行った後，各グループごとのプロファイル結果から内部動作を調査したものである．図中，1 ブロックは 5 秒間に相当する．この結果から，本アプリケーションの処理内容は，時間方向に対して大きく（1）初期処理部，（2）本計算部，（3）後処理部にグルー

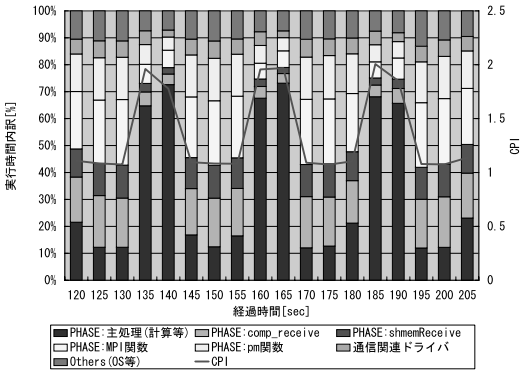


図 9 PHASE 本計算部の時系列の関数プロファイル結果と CPI の変化
Fig. 9 Behavior of time-series profiling results and CPI during the main calculation part of PHASE.

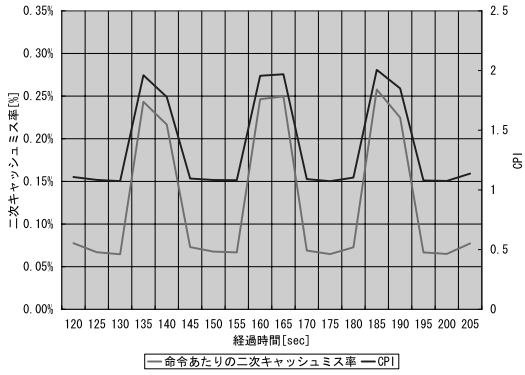


図 10 PHASE 本計算部の CPI および 2 次キャッシュミス率の変化
Fig. 10 Behavior of CPI and second level cache miss ratio during the main calculation part of PHASE.

プ化されていることが分かる．さらに，この分析結果から各処理時間を比較すると，本計算部の並列化効率が悪いことが分かる．8 ノードの場合に後処理部のグループがなくなっているのは，当該処理の実行時間が数秒と短くなり，本計算部分と同一グループに分類されてしまったためである．

以上のように，時間方向・ノード方向の両方向にクラスタ分析を適用することにより，PHASE の実行全体を処理内容ごとに分類するとともに，全ノードのの動作内容がほぼ同じであることを確認できた．本分析結果についても 5.2 節と同様に，同じ動作を行う計算ノードを簡略表示することでコンパクトな画面表示も実現できる．

5.4 時間方向のクラスタ分析と CPU 性能情報の統合分析効果

本節では，高速化に重要となる本計算部に対して，時間方向の時系列データに対するクラスタ分析と CPU 性能情報を組み合わせることによるチューニングについて説明する．

5.3 節において抽出した本計算部のプロファイル結果および CPI を**図 9** に示す．このグラフは，本計算部中の 90 秒間について拡大表示したものである．実際に，iScopes でクラスタ分析を行うと，プロファイル結果に従って，2 グループの繰返しを検出できる．本計算部には一定の周期性があり，計算処理が主である 10 秒間と通信処理が主である 15 秒間とに分類できる．

まず，CPU 性能がボトルネックとなっている計算処理を高速化する場合に CPU 性能情報を組み合わせる．同時間帯の CPI および 2 次キャッシュミス率を表示したグラフを**図 10** に示す．この図から分かるように，当該区間では，2 次キャッシュミス率が 0.07%から 0.27%まで悪化すると同時に，CPI が 1.0 から 2.0 に悪化していることが分かる．この結果に基づき，ユーザはキャッシュメモリ関連のチューニングを選択することができる．

次に，通信処理がボトルネックとなっている部分については，プロファイル結果から PHASE プログラム内部での受信待ち処理が多いことが分かる．CPI が 1.0 前後でありキャッシュミス率も低いことから，高速化のためには PHASE プログラム内部の受信待ち処理を簡素化する（たとえば命令数を減らす），もしくはハードウェア通信性能を向上させるという選択が可能となる．

gprof に代表される，アプリケーションの開始から終了までの全体に対して，関数ごとの消費時間や CPU 性能情報を一括して計測するだけでは上記のような周期性は見出すことができず，また，区間ごとの CPU 情報と組み合わせたチューニングを行うこともできない．さらに，iScopes は，gprof では計測できない OS や通信ドライバ部分についてもその消費時間も含めて計測できる．

以上のように，iScopes のプロファイリング・CPU 性能情報・クラスタ分析による統合解析手法は，異常発見だけでなく並列処理効果解析等のアプリケーションの効率的な調査にも有用である．PHASE のような周期性をともなうアプリケーションについて，時間方向のクラスタ分析によって全データを少数のグループに分類でき，分析対象を大幅に減少させることができる．5.2 節で前述した姫野ベンチマークと同様に，クラスタ規模が大きくなった場合でも，クラスタ分析によるグループ数が大幅に増加することはないと予想され

る．したがって，本分析手法を同様に適用可能である．

6. 関連研究

広く利用されているプロファイリングツールとして GNU gprof[2] がある．gprof は，PMC を用いず，関数呼び出し回数を計測するコードを測定対象プログラム内に直接埋め込む．したがって，iScopes のように複数のアプリケーションや OS・ライブラリを同時に測定することはできず，また，PMC を用いないため CPI 等の CPU 性能情報は計測できない．一方，PMC を用いたプロファイリングツールとしてオープンソースソフトウェアとして OProfile[3]，商用アプリケーションとして Intel VTune Performance Analyzer[12] があるが，これらのツールは測定データの記録方法が iScopes と異なるため，低いオーバヘッドでの時系列データを採取することはできない．

多数台の計算ノードを対象とした分析ツールとして PerfExplorer[4] が提案されている．PerfExplorer は，ノード方向に対するクラスタ分析を行っており，その分析方法も iScopes と同じく階層型クラスタ分析を用いている．実際に 8000 ノード以上で構成された BlueGene/L を使用して検証を行っており，クラスタ分析の実システムにおける有用性を確認している．しかしながら，PerfExplorer の目的にはシステム全体を測定対象とする障害解析や性能分析は含まれていない．そのため，OS やデバイスドライバ部分には言及していない．加えて時間方向のクラスタ分析を行っていないため，iScopes の特長である周期性の発見や分析には不向きである．

また，同様に多数台の計算ノードを対象とした性能評価・分析を行うための手法として，Vetter らは各計算ノードのデータ（MPI ログおよび PMC 情報）を統計解析により分類し，性能ボトルネックを抽出する手法を検討している[13)~17)]．しかしながら，これらの研究は，時系列データの解析に関して言及しておらず，また，様々な統計手法の検討にとどまっており iScopes のように実システムに適用したうえでその効果を確認できた例は報告されていない．

並列分散システムを対象として，大量の測定データを処理し可視化するツールとして，Paradyn[18] がある．Paradyn は，W^3 サーチモデル（why，when，where）を提唱しており，性能ボトルネックの原因（why）が，「いつ（when）」「どこで（where）」発生しているかを特定することを目的としている．Paradyn は，SHG（Search History Graph）と呼ぶ CPU 実行時間の長い関数群を選択し，ツリー構造を構成することで解析を行っている．Roth らはさらに，SHG をコンパクト表示するための SGFA（Sub-Graph Folding Algorithm）を提案している[19]．これらの研究の性能ボトルネックに対するアプローチは，本論文で提案した解析手法と同様である．しかしながら，これらの研究は，単一対象アプリケーションの関数実行時間のみに着目しており，本論文で述べたような時系列関数プロファイリングや CPU 性能情報との統合には至っていない．

PC クラスタを対象として特に通信部分の分析に重点をおいたツールに Intel Trace Analyzer and Collector[20] がある．このツールは，プログラムの時系列の挙動を表示可能であるが，統計処理を導入していないので，iScopes が提供するクラスタ分析によるデータの簡略表示等を行うことはできない．また，CPU 性能情報を同時に分析対象とすることもできない．

SMP マシン（IBM SP-2）を対象とした同様の研究として文献 21) があげられる．この研究は MPI ログのみを対象としており，MPI の種類別所要時間内訳を時系列で計測可能である．*jumpshot* と呼ばれる可視化ツールを含んでおり，当該データの可視化も可能である．しかしながら，統計処理機能を含んでいないため可視化に際して大規模システムに適用することは困難である．また，iScopes は MPI 以外の関数情報や CPU 性能情報も出力可能である．

7. おわりに

本論文では，PC クラスタに対する新たなシステム解析手法を提案した．時系列データを対象としたクラスタ分析により，これまで困難とされていたシステムの異常動作の原因特定やアプリケーションの時間方向での動作分析，大量の性能情報の効率的な可視化が可能となった．実際に，本手法を実装した統合性能解析ツール iScopes を開発し，実システムに適用することでその有用性を確認できた．

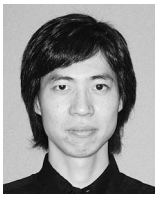
本論文では，姫野ベンチマークおよび PHASE を例として分析を行った．これらのアプリケーションは，HPC アプリケーションの特性である，同じ関数やループ処理を繰り返し実行するため，iScopes の時間方向のクラスタ分析により定常状態や周期性を効率良く分析できた．しかしながら，このような特性を持たない HPC アプリケーションの場合には，その適用が困難である．また，クラスタ分析によるグループ数の指定をユーザが決定する必要があり，グループ数が多くなる場合には効率的な分析は行えない．今後は，特にグループ数の決定手法の観点から，他のアプリケーションに対して適用し改良を行う．

また，今後は，iScopes の大規模な PC クラスタへの実適用を行いその効果を確認する予定である．加えて，シングルユーザ環境だけではなく，マルチユーザ環境における障害解析や性能問題にも適用を進める．本ツールは PC クラスタ以外の，大量の計算ノードを必要とする同等システム（グリッドシステムや多数台の Web フロントエンド等）にも適用可能である．この点についても今後検証を行う．

参 考 文 献

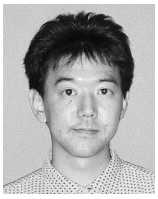
1) RSCC：理研スーパーコンバインドクラスタ. http://accc.riken.jp/rscc/
2) GNU gprof. http://www.gnu.org/software/binutils/manual/gprof-2.9.1/gprof.html
3) OProfile. http://oprofile.sourceforge.net/
4) Huck, K.A. and Malony, A.D.: PerfExplorer: A Performance Data Mining Framework for Large-Scale Parallel Computing, *Proc. 2005 ACM/IEEE conference on Supercomputing*, Published by CD-ROM (2005).
5) 姫野ベンチマーク. http://accc.riken.jp/HPC/HimenoBMT/
6) IA-32 Intel Architecture Software Developer's Manual Volume 3: System Programming Guide.
7) Browne, S., Dongarra, J., Garner, N., London, K. and Mucci, P.: A Scalable Cross-Platform Infrastructure for Application Performance Tuning Using Hardware Counters, *Proc. 2000 ACM/IEEE conference on Supercomputing*, Published by CD-ROM (2000).
8) 神嶌敏弘：データマイニング分野のクラスタリング手法（1），人工知能学会誌，Vol.18, No.1, pp.59–65 (2003).
9) 神嶌敏弘：データマイニング分野のクラスタリング手法（2），人工知能学会誌，Vol.18, No.2, pp.170–176 (2003).
10) SCore Cluster System Software. http://www.pccluster.org/
11) 文部科学省 IT プログラム戦略的基盤ソフトウェアの開発ナノシミュレーション PHASE Ver.4.0. http://www.fsis.iis.u-tokyo.ac.jp/result/software/
12) Intel VTune Performance Analyzer. http://www.intel.com/cd/software/products/asmo-na/eng/vtune/index.htm
13) Ahn, D.H. and Vetter, J.S.: Scalable Analysis Techniques for Microprocessor Performance Counter Metrics, *Proc. 2002 ACM/IEEE conference on Supercomputing*, Published by CD-ROM (2002).
14) Vetter, J.S.: Dynamic Statistical Profiling of Communication Activity in Distributed Applications, *Proc. 2002 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*, pp.240–250 (2002).
15) Vetter, J.S. and McCracken, M.O.: Statistical Scalability Analysis of Communication Operations in Distributed Applications, *Proc. 8th ACM SIGPLAN symposium on Principles and practices of parallel programming*, pp.123–132 (2001).
16) Vetter, J.S. and Reed, D.A.: Managing Performance Analysis with Dynamic Statistical Projection Pursuit, *Proc. 1999 ACM/IEEE conference on Supercomputing*, Published by CD-ROM (1999).
17) Vetter, J.S.: Performance Analysis of Distributed Applications using Automatic Classification of Communication Inefficiencies, *Proc. 2000 ACM/IEEE International Conference on Supercomputing*, pp.245–254 (2000).
18) Miller, B.P., Callaghan, M.D., Cargille, J.M., Hollingsworth, J.K., Irvin, R.B., Karavanic, K.L., Kunchithapadam, K. and Newhall, T.: The Paradyn Parallel Performance Measurement Tool, *IEEE Computer*, Vol.28, No.11, pp.37–46 (1995).
19) Roth, P.C. and Miller, B.P.: On-line Automated Performance Diagnosis on Thousands of Processes, *ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming* (*PPoPP*), pp.69–80 (2006).
20) Intel Trace Analyzer and Collector. http://www.intel.com/cd/software/products/asmo-na/eng/cluster/tanalyzer/index.htm
21) Wu, C.E., Bolmarcich, A., Snir, M., Wootton, D., Parpia, F., Chan, A., Lusk, E. and Gropp, W.: From Trace Generation to Visualization: A Performance Framework for Distributed Parallel Systems, *Proc. 2000 ACM/IEEE conference on Supercomputing*, Published by CD-ROM (2000).

(平成 18 年 1 月 27 日受付)
(平成 18 年 4 月 30 日採録)



山村 周史（正会員）

1998 年京都工芸繊維大学大学院電子情報工学科修士課程修了．2001 年同大学院情報・生産科学専攻博士課程修了．博士（工学）．同年富士通（株）入社．現在（株）富士通研究所勤務．並列計算機アーキテクチャ，プロセッサアーキテクチャに関する研究に従事．PC クラスタ，Linux システムの性能評価・チューニングに興味を持つ．電子情報通信学会，IEEE 各会員．

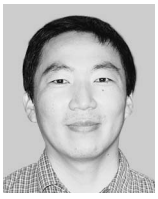


平井 聡（正会員）

（株）富士通研究所勤務．1997 年より同研究所にて IA プロセッサを使用した大規模 PC サーバおよび PC クラスタシステム向け性能向上技術の研究に従事．現在，Linux システムをベースとした高性能，高信頼技術の研究を行っている．

小野美由紀（正会員）

（株）富士通研究所勤務．オブジェクト指向型言語処理系，マルチメディアデータベース，データマネジメント等の研究開発に従事．現在，Linux システムに対する性能評価ツールの GUI の研究開発を行っている．

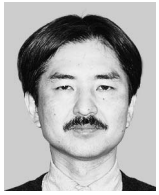


松本 和宏

（株）富士通研究所勤務．PC クラスタにおけるパラメータ最適化に関する研究開発に従事．計算機性能データを対象とした統計解析やデータマイニングに関心を持つ．

住元 真司（正会員）

1986 年同志社大学工学部電子工学科卒業．同年富士通（株）入社．(株）富士通研究所にて並列オペレーティングシステム，並列分散システムソフトウェアの研究開発に従事．1997 年より新情報処理開発機構に出向．コモディティネットワークを用いた高速通信機構の研究開発，RWCP SCore2，SCore3 クラスタ等大規模 PC クラスタ開発に従事．2002 年より（株）富士通研究所にて高速通信機構の研究開発，理研スーパーコンバインドクラスタ等大規模 PC クラスタ，UHPC システムの開発等に従事，並列分散システムのアーキテクチャ，システムソフトウェア等に興味を持つ．平成 12 年度情報処理学会論文賞受賞，工学博士（慶應義塾大学大学院理工学研究科）．



久門 耕一（正会員）

1979 年東京大学工学部電気工学科卒業．1981 年同大学大学院電子工学専門課程修士課程修了．1984 年同大学院博士課程中退．同年（株）富士通研究所入社．現在，同社 IT コア研究所に所属．CPU，メモリ，並列計算機アーキテクチャに関する研究に従事．GCC，Linux カーネル等の改良にも興味を持つ．