情報処理学会論文誌 Vol.59 No.2 362–371 (Feb. 2018)

仮想化を利用した設備機器の状態監視における計測時刻誤差の推定手法

金子 雄[1,a)] 伊藤 俊夫[1,b)] 原 隆浩[2,c)]

受付日 2017年5月9日, 採録日 2017年11月7日

概要：ビルの設備機器を遠隔から管理する遠隔ビル管理システムには，設備機器の状態を計測するクローラと呼ばれるソフトウェアが存在する．計算機の仮想化（VM; Virtual Machine）を利用し，各クローラをVM 上で実行することで，クローラの障害の波及範囲を制限しつつ，遠隔ビル管理システムに必要な物理マシン数を減らせる．多くのクローラを単一の物理マシン上で実行すると，計測のタイミングが乱れやすくなるため，計測時刻の誤差を許容できる範囲で，多くの VM を実行することが重要となる．本論文では，物理マシンや VM のモデルに基づくシミュレーションにより，クローラの計測時刻の誤差を推定する方法を提案する．提案手法による推定値と，実機による実測値とを比較し，提案手法の推定精度を検証する．

キーワード：監視制御アプリケーション，クローラ，仮想化，Xen

Estimation Method for Measurement Time Errors of Virtualized Building Facilities Monitoring Applications

Yu Kaneko[1,a)] Toshio Ito[1,b)] Takahiro Hara[2,c)]

Received: May 9, 2017, Accepted: November 7, 2017

Abstract: In a remote building management system (BMS) that manages facilities of buildings, there is an application called a *crawler* that measures statuses of facilities via a network. By running a crawler on a Virtual Machine (VM) for each building owner, the number of physical machines in the remote BMS can be reduced. However, running multiple crawlers on the same physical machine causes measurement time fluctuation, which may result in inaccurate monitoring and control of facilities. We propose an estimation method for measurement time accuracy of crawlers according to simulation with a model of crawlers, VMs, and physical machines. The relationship between the number of crawlers running on the same physical machine and their measurement time accuracy can be determined by the proposed method. We verify the estimation accuracy of the proposed method by comparing with measurement time accuracy of real machines.

Keywords: monitoring and control applications, crawler, hardware virtualization, Xen

1. はじめに

インターネットなどの Wide Area Network（WAN）を経由して，複数のビルに設置された設備機器を管理する遠隔ビル管理システム（BMS; Building Management System）がある [1]．従来の BMS のようにビル内に専用の監視室を設ける必要がないため，中規模・小規模のビルへ適用しやすい．遠隔 BMS の管理対象は，空調や照明などの設備機器の稼働状態や，室内の温度や湿度などである．以降，これらの管理対象のことを「監視点」，また監視点から得られるデータのことを「監視点データ」と呼ぶ．

遠隔 BMS の監視制御機能は通常の BMS と類似しており，たとえば以下である．

[1] 株式会社東芝研究開発センター
Research & Development Center, Toshiba Corporation, Kawasaki, Kanagawa 212–8582, Japan
[2] 大阪大学大学院情報科学研究科
Graduate School of Information Science and Technology, Osaka University, Suita, Osaka 565–0871, Japan
a) yu1.kaneko@toshiba.co.jp
b) toshio9.ito@toshiba.co.jp
c) hara@ist.osaka-u.ac.jp

- **可視化機能**：監視点データを，ビル管理者に分かりやすく表示する機能．最新値を表示したり，値の推移をグラフとして表示したりする．
- **課金機能**：監視点データから設備機器の稼働時間を計算し，テナントに対する課金額を計算する機能．
- **フィードバック制御機能**：ある設備機器に対してフィードバック制御を行う機能．たとえば，バルブを調節することで空調用の冷却水の流量を一定に保つ処理がある．

遠隔 BMS には，ビル群の監視点データを計測し，上記の監視制御機能に提供するための機能が存在する [2]．以降，本機能のことを「クローラ」と呼ぶ．

遠隔 BMS が管理する各ビルにはオーナがいる．ビルオーナごとに専用の物理マシン（PM; Physical Machine）を用意し，各 PM 上でクローラを実行することで，あるクローラに障害が発生しても，他のビルオーナへの監視制御機能の提供を継続できる．しかしこの構成は，PM やその電気代，PM が占有する物理スペースに対する設備投資を増大させる．

クローラの障害の影響範囲を制限しつつ，必要な PM 数を減らすための方法として，計算機の仮想化技術がある [3], [4]．仮想化技術により，PM 上に複数の仮想的な計算機（VM; Virtual Machine）を作成できる．各 VM は個別の Operating System（OS）を実行できるため，OS やクローラの障害の影響を，それを実行する VM 内に制限できる．

設備投資を節約するためには，各 PM 上で，できるだけ多くのクローラ/VM を実行したい．しかし，VM 上で動作するアプリケーションの性能は，複数の VM の並列実行により低下する [5], [6]．そのため，多くのクローラを単一の PM 上で実行すると，監視点データの計測時刻の誤差が大きくなり，監視制御機能の正常な動作を阻害する．したがって，計測時刻誤差が許容値を超えない範囲で，できるだけ多くのクローラを実行できるとよい．

そこで我々は，VM 上で動作するクローラの計測時刻誤差を推定する手法を提案する．本手法は，PM や VM のモデルに基づくシミュレーションにより，複数のクローラ/VM を単一の PM 上で実行した場合の計測時刻誤差を推定する．遠隔 BMS の運用者は，この推定結果と，計測時刻誤差の許容値から，PM 上で実行できるクローラ/VM の最大数を得ることができる．

以降，2 章で関連研究を説明し，3 章で遠隔 BMS の概要を述べる．4 章で提案手法を説明する．5 章で提案手法による推定の結果を実機評価の結果と比較し，提案手法の推定精度を評価する．最後に 6 章でまとめる．

2. 関連研究

近年，仮想化技術の成熟にともない，監視制御アプリケーションを VM 上で実行することを想定した研究が行われている [7], [8], [9]．VM よりもオーバヘッドが小さい仮想化技術としてコンテナ [10] があるが，本論文では，高可用性の観点で優位な VM に着目する [11]．

監視制御アプリケーションは性能要件が厳しいため，VM を適用した場合の性能の低下に注意すべきである．複数の VM を単一の PM 上で実行する場合，アプリケーションの性能は低下する [12], [13]．この性能低下を考慮し，それでもアプリケーションの性能要件を満たせると判断できた場合に限り，PM に VM を追加するべきである．

PM のリソース残量に基づいて，アプリケーションの性能要件を満たせるかを判断する手法がある．このリソースベースの手法は，追加する VM のリソース消費量の予測値よりも，PM のリソース残量が大きい場合に，VM の追加を許可する [14], [15], [16]．しかし監視制御アプリケーションは，PM のリソースが余っていても，VM 間の干渉により性能要件を満たせない場合があるため [12]，リソースベースの手法では性能要件の達成が難しい．

DeepDive [17] は，追加する予定の VM と類似する負荷を PM に与えることで，稼働中のアプリケーションの性能が低下するかを確認する．稼働中の PM に負荷を与えるため，性能低下を正確に把握できるが，稼働中のアプリケーションの性能を低下させるリスクが生じる．文献 [18] で提案されている手法は，稼働中の VM をコピーすることで製品環境と同じテスト環境を構築し，テスト環境で性能を評価する．この手法は稼働中のアプリケーションの性能に影響を与えないが，テスト用のサーバやネットワーク機器，設備機器などへの設備投資が必要となる．

文献 [19] で提案されている手法は，事前の性能評価の結果に基づいて，アプリケーション追加時に性能が低下するかを判断する．この手法は，稼働中のアプリケーションに影響を与えず，かつ，設備投資を抑えられるが，どのくらい性能が低下するかを推定できない．文献 [20] で提案されている手法は，VM 間の性能干渉を考慮して，VM の性能変化を定式化している．文献 [20] では，円周率演算の速度を性能として，提案の式の正しさを検証している．しかし，本論文が想定するクローラの性能（計測時刻の正確さ）の変化は，文献 [20] の式にあてはまらなかった．クローラの挙動は通信処理を含み，通信処理は VM 群を管理する仮想化ソフトウェアにも負荷を与えるため，処理が VM 内で完結する円周率演算と比べて，性能の定式化が難しいと考える．そこで我々は，クローラや VM の処理をシミュレーションにより模擬することで，クローラの計測時刻誤差を推定する方法を提案する．

3. 遠隔ビル管理システムの概要

3.1 クローラによる機器状態の計測

図 1 は，遠隔 BMS とビル群のシステム構成である．遠

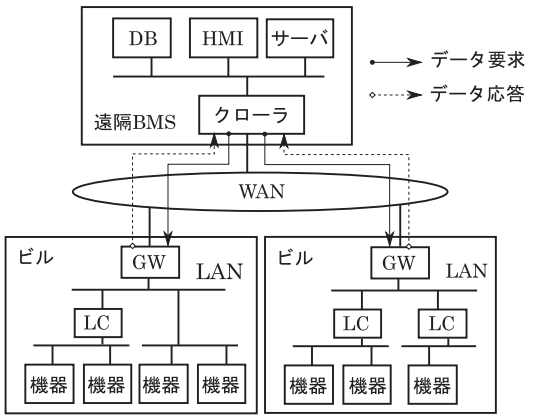


図 1 遠隔 BMS とビル群のシステム構成
Fig. 1 System architecture of a remote BMS and buildings.

隔 BMS はクローラや，監視点データを保存するデータベース（DB），監視制御機能が動作するサーバ，運用者が操作する Human Machine Interface（HMI）を備える．図ではそれぞれ 1 つだけだが，実際には複数個が存在しうる．遠隔 BMS とビル群は WAN で接続される．

クローラは，監視点データを取得するため，ビルに設置されたゲートウェイ（GW）にデータ要求を送信する．クローラと GW は，BACnet/WS [21] や IEEE1888 [22] などの遠隔監視制御向けプロトコルを使い監視点データを送受信する．GW はデータ要求を受けると，ローカルコントローラ（LC; Local Controller）または機器から監視点データを取得する．GW はデータ応答として監視点データをクローラに返し，クローラは監視点データを DB に蓄積する．

本論文では，クローラから通信を開始するリクエスト/レスポンス型の通信パターンを想定する．ビル側から遠隔 BMS に対して監視点データを送信する通信パターンも考えられるが，両者は一長一短である．リクエスト/レスポンス型の通信パターンの長所は，いつ，どの監視点データを計測するかという設定情報を，遠隔 BMS で集中管理できる点である．そのため，設定情報の変更が容易であり，GW や LC などのビル側の装置を簡素化できる．この長所は遠隔 BMS の適用を容易にすることにつながると考え，リクエスト/レスポンス型を想定する．

3.2 クローラの要件

クローラは，事前に定められた時刻に監視点データを計測し，監視制御機能に提供する必要がある．この事前に定められた時刻と，クローラが実際に計測した時刻の差を，計測時刻誤差と呼ぶ．計測時刻誤差は小さいほうがよい．以下にその理由を述べる．

- 可視化機能は，監視点データが正時に正確に計測されていれば，ビル管理者にとって見やすいグラフを表示できる．
- 課金機能は監視点データから機器の稼働時間を計算し，テナントへの課金額を計算する．そのため，一定間隔で稼働または非稼働を判定することが重要である．
- 監視点データが計画どおりに計測されないと，フィードバック制御機能の品質が低下する．たとえば，制御対象の機器の状態が不安定になる．

許容できる計測時刻誤差は，監視制御機能の種類や，目指す品質に依存する．たとえば，可視化機能や課金機能は 1 秒間隔で監視点データを計測する場合があるため，1 秒の誤差は許容できない．またフィードバック制御機能は，50〜100 ミリ秒の誤差を許容できない場合がある．このように，計測時刻誤差を許容範囲内で抑えることが，クローラの性能要件である．

またクローラは，性能要件を高い確率で達成する必要がある．本論文では，IEC61508 Safety Integrity Level の連続動作モードのレベル 1 に基づき，99.999%を目標とする．SIL1 は「故障しない確率」を 99.999%以上と定義しているが，本論文では「性能要件を満たせない状態」も「故障」と見なす．監視制御システムでは，性能要件を満たせないことは，異常事態と同等だからである．以上より，99.999%以上の確率で，計測時刻誤差を許容範囲内で抑えることを，クローラの要件とする．

厳密には，計測時刻誤差がゼロでも，クローラとビルの間の WAN における通信遅延のばらつきや，ビル側の装置における処理時間のばらつきが存在する．しかし，通信品質を保証可能な WAN 回線を利用すれば，通信遅延のばらつきは抑えられる．また，ビル側の装置は，監視制御システムで使用される機器であるから，そもそも処理時間のばらつきは小さい．また，本論文では，計測を行うクローラと，制御を行うフィードバック制御機能とを，別のアプリケーションとして実装することを想定するため，クローラにおいては，計測のための通信処理のみを考慮すればよい．以上より，本論文では，クローラの計測処理時間のばらつき（計測時刻誤差）に着目し，性能指標とする．

3.3 クローラを追加するユースケース

クローラは，ビルオーナと遠隔 BMS 事業者との間でビル管理契約が締結された場合に追加される．ビル GW の IP アドレスや通信プロトコル，各監視点の識別子や計測時刻などを DB に登録してから，クローラを追加する PM を選択する．そして選択された PM 上に VM を作り，クローラを実行する．PM ごとにクローラを追加した場合の計測時刻誤差を推定できれば，性能要件を満たせる PM を適切に選択できる．

ビル管理契約の締結には人が介在するため，遠隔 BMS の運用者は，クローラ/VM を追加するタイミングを調節できる．仮に同時に複数のビル管理契約が成立したとしても，クローラの追加は逐次的に行える．別のいい方をするなら，クローラを同時に追加することはない．また，1 度

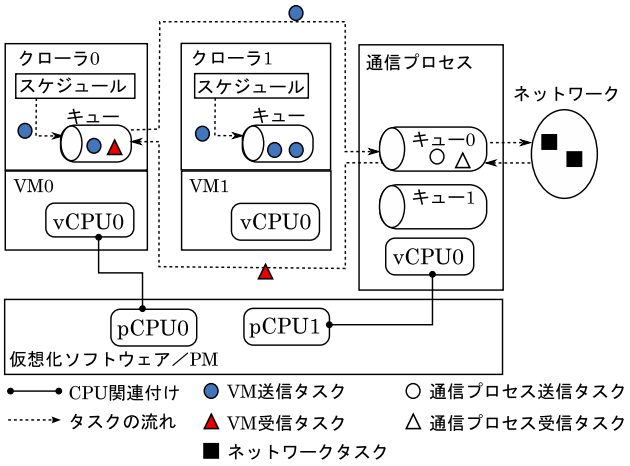


図 2 シミュレーションのモデル
Fig. 2 Model of crawlers, VMs and physical machines.

追加したクローラを，別の物理マシンに移動することは考えない．VM のライブマイグレーションを使えば，クローラを停止することなく別の物理マシンへ移動できるが，移動中はクローラの性能が低下するためである．

4. 提案手法

提案手法は，クローラや VM，物理マシンのモデルに基づくシミュレーションにより，計測時刻誤差を推定する．

4.1 シミュレーションのモデル

監視点データの計測処理は，送信処理と受信処理に分割できる．また，クローラ/VM と仮想化ソフトウェアの双方が，両方の処理に関わる．提案手法は，この粒度で，クローラや仮想化ソフトウェアの挙動をモデル化する．詳細な挙動をモデル化しないことで，異なるアプリケーションや仮想化ソフトウェアに対しても適用しやすい手法を目指す．

図 2 に，シミュレーションのモデルを示す．以下，各要素を説明する．

- **仮想化ソフトウェア/PM** は，1 個以上の物理 CPU（pCPU）を持つ．仮想化ソフトウェアは，pCPU と仮想 CPU（vCPU）との関連付けを管理する．クローラの処理は CPU バウンドであるため，メモリやストレージなどは考慮しない．
- **VM** は，1 個以上の vCPU と 1 個のクローラを持つ．vCPU の状態遷移は 4.3 節で説明する．
- **クローラ**は，監視点データを計測するスケジュール（計測スケジュール）とタスクキューを持つ．計測スケジュールは，いつ，どの監視点を計測するかを定めている．クローラの計測処理は「タスク」としてモデル化する．タスクはタスクキューに格納され，vCPU により処理される．タスクの詳細は 4.2 節で説明する．
- **通信プロセス**は，物理的な NIC を使用し，別の物理マシンとの通信を行うソフトウェアである．実際には仮想化ソフトウェアの一部として実装される場合もあり，たとえば Xen の Dom0 や，KVM のホスト OS に相当する．通信プロセスの通信処理もタスクとしてモデル化する．通信プロセスは 1 個以上の vCPU と，VM ごとに有限サイズのタスクキューを持つ．
- **ネットワーク**は，データ要求をビル GW に送信してから，データ応答が返ってくるまでの応答時間を模擬するためのタスクプールである．

4.2 監視点データの計測処理のモデルの詳細

前節で述べたように，提案手法では，監視点データの計測処理を送信処理と受信処理という 2 つの処理に単純化してモデル化する．クローラ/VM における両処理を，VM 送信タスク（ST_{vm}）と VM 受信タスク（RT_{vm}）としてモデル化する．ST_{vm} は，データ要求の作成や，データ要求を通信プロセスに渡す処理に相当する．RT_{vm} は，データ応答を受信して解析する処理に相当する．どちらのタスクも，タスクを処理するために必要な時間を表すパラメータ T_{vm} を持つ．

同様に，通信プロセスにおける両処理を，通信プロセス送信タスク（ST_{cp}）と通信プロセス受信タスク（RT_{cp}）として定義する．どちらのタスクも，タスクを処理するために必要な時間を表すパラメータ T_{cp} を持つ．また，ビル GW の応答時間をネットワークタスク NT として定義する．NT も，タスクを処理するために必要な時間 T_{net} を持つ．

つまり，ある監視点データの計測処理を，ST_{vm}，ST_{cp}，NT，RT_{cp}，RT_{vm} の 5 つのタスクとしてモデル化する．これらのタスクは，以下のようにシミュレーションされる．

(1) クローラの計測スケジュールに従い，ST_{vm} が作成され，タスクキューに格納される．
(2) ST_{vm} の処理が完了すると，通信プロセスのタスクキューに ST_{cp} を格納する．ただし，タスクキューが一杯の場合は格納できない．ST_{vm} は削除される．
(3) 処理が完了した ST_{cp} はタスクキューから削除される．そして，代わりに NT が生成され，ネットワークのタスクプールに格納される．
(4) 時間 T_{net} が経過すると，NT は削除される．そして RT_{cp} が通信プロセスのタスクキューに格納される．
(5) RT_{cp} の処理が完了すると，RT_{cp} は削除され，代わりに RT_{vm} が対応する VM のタスクキューに格納される．
(6) RT_{vm} の処理が完了すると，RT_{vm} は削除される．これをもって監視点データの計測処理は完了となる．

以上の処理において，ST_{vm} が VM のタスクキューに挿入された時刻と，その ST_{vm} がタスクキューから削除された（処理が完了した）時刻の差を，計測時刻誤差とする．ST_{cp} の処理が完了した時刻を用いて計測時刻誤差を計算することも可能だが，今回の評価で用いるクローラは，ク

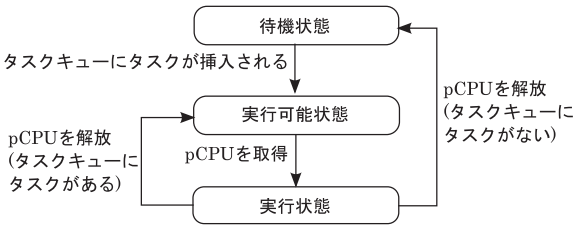


図 3 モデル化した仮想 CPU（vCPU）の状態遷移
Fig. 3 State machine of model of virtual CPUs.

ローラにおける送信処理が完了した時刻を用いて計測時刻誤差を計算するため，それに合わせる．

4.3 物理 CPU と仮想 CPU のモデルの詳細

PM のリソースが余っていても，VM 間の CPU 競合によりアプリケーションの性能は低下する [12]．CPU 競合とは，pCPU を使用したい VM の vCPU が，pCPU を使用できない状態のことである．ある時点において，「pCPU を使用したい VM の vCPU の数」が「PM が持つ pCPU の数」を超えると，CPU 競合が生じる．CPU 競合はクローラの計測時刻誤差の主要因であるため，これを模擬できるように pCPU と vCPU をモデル化する．

図 3 はモデル化した vCPU の状態遷移図である．vCPU の初期状態は「待機状態」である．1 つ以上の ST_{vm} または RT_{vm} がタスクキューにあるとき，そのクローラを実行する VM の vCPU は，「実行可能状態」となる．pCPU は，実行可能状態である vCPU をランダムで 1 つ選ぶ．pCPU に選ばれた vCPU は，タイムスライスが割り当てられ，「実行状態」となる．タイムスライスとは，pCPU を解放することなく使用し続けられる時間である．vCPU に割り当てるタイムスライスの量については，4.4.2 項で述べる．

実行状態の VM の vCPU は，タスクキューから First-In First-Out（FIFO）でタスクを選ぶ．そして，自身に割り当てられたタイムスライスを消費し，消費した分だけ，タイムスライスを消費することでタスクを処理する．タスクは，処理された時間が T_{vm} に達すると，処理完了となる．ここで，タスク（VM 送信タスクと VM 受信タスク）の処理完了に必要な時間は，等しく T_{vm} とする．pCPU に選ばれなかった vCPU は，タスクを処理できない．いずれかの pCPU が解放され，次に自身が選ばれることを待つ必要がある．この待ち時間が CPU 競合を表すことになる．

同様に，実行状態の通信プロセスの vCPU は，いずれかのタスクキューから，FIFO でタスクを取得し，処理する．タスクは，処理された時間が T_{cp} に達すると，処理完了となる．

pCPU に選ばれた vCPU は，以下のいずれかの条件を満たした場合に，pCPU を解放する．

- タスクキューにタスクが存在しない場合
- 自身に割り当てられたタイムスライスを使いきった場合

pCPU を解放した vCPU は，タスクキューにタスクがあるなら実行可能状態に，タスクがないなら待機状態に遷移する．解放された pCPU は，再び，実行可能状態の vCPU を選ぶ．

4.4 モニタリング情報に基づくパラメータの決定

タスクの処理にかかる時間や，vCPU が pCPU を解放するまでの時間は，PM の性能に依存する．実機を用いたテスト環境を構築し，十分な評価を行うことで，これらのパラメータを正確に求められるかもしれないが，そのためにはテスト用のサーバやネットワーク機器，設備機器に対する設備投資が必要となる．そこで提案手法では，実際にビル群に対して稼働している PM（製品環境の PM）から得られる情報を用いて，これらのパラメータを決定する．

4.4.1 タスクの処理時間の決定

T_{vm} と T_{cp} を決定するために，製品環境の PM で，一定期間 D ごとに，以下のデータを計測する．

- U_{vm}：PM 上の VM 群の CPU 使用時間の合計（秒）
- U_{cp}：通信プロセスの CPU 使用時間（秒）
- N：期間 D において，PM 上のクローラ群が計測した監視点データの合計

CPU 使用時間は仮想化ソフトウェアが提供するコマンドにより計測できる．たとえば Xen であれば `xentop` コマンドや `xl` コマンドで計測できる．

T_{vm} は，以下のように計算する．

$$T_{vm} = \frac{U_{vm}}{N} \times \frac{1}{2} \tag{1}$$

今回，送信タスクと受信タスクの処理にかかる時間は等しく T_{vm} という前提をおいている．そのため，1 回の計測処理にかかる時間（$\frac{U_{vm}}{N}$）を半分にした値を，T_{vm} とする．

同様に，T_{cp} は以下のように計算する．

$$T_{cp} = \frac{U_{cp}}{N} \times \frac{1}{2} \tag{2}$$

また，監視点データを計測するごとに，ビル GW の応答時間を計測し，その平均値を T_{net} とする．

4.4.2 タイムスライスの決定

製品環境の PM で，vCPU が pCPU を解放するまでの時間を計測し，それをタイムスライスとする．Xen であれば `xentrace` コマンドを使用することで，この時間を計測できる．ただし，vCPU が pCPU を解放するまでの時間は，一定ではなく，状況により変化する．`xentrace` コマンドによれば，vCPU が pCPU を解放するまでの時間と，その発生頻度の組合せを取得できるため，それを線形補間したものを，タイムスライスの割当て量を決定するための確率密度関数とする．

5. 評価

提案手法の有効性を検証するために，実機の計測時刻誤

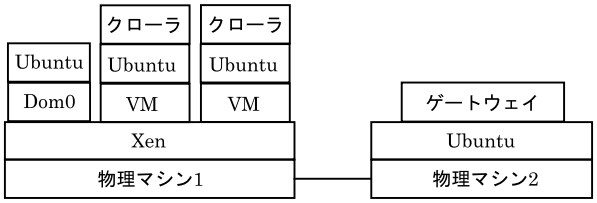


図 4 実機評価の環境
Fig. 4 Experimental environment using real machines.

差と，提案手法により推定した計測時刻誤差を比較する．

5.1 実機評価の環境

実機評価の環境を**図 4** に示す．評価には 2 つの PM を用いる．1 つはクローラを実行するために，もう 1 つはビル GW 群と監視点を模擬するために用いる．2 つの PM は 1000BASE-T で接続する．クローラ用の PM は Intel(R) Xeon(R) 1.80 GHz CPU（8 コア），32 GB メモリを備える．仮想メモリアドレスと物理メモリアドレスの変換処理を効率化する Extended Page Tables 機能は有効化する．ビル GW 用の PM は Intel(R) Xeon(R) 2.60 GHz CPU（32 コア），80 GB メモリを備える．PM と VM の OS として Ubuntu 16.04 LTS（Linux 4.4.0）を使用する．

仮想化ソフトウェアとして Xen [4] を採用する．Xen は，動作の安定性とメンテナンス性を重視した設計となっており，監視制御アプリケーションに適している [23]．バージョン 4.6.0 の Xen を使用し，デフォルトのスケジューラである credit スケジューラを使用する．credit スケジューラの timeslice パラメータは，デフォルト値である 30 ミリ秒を使用する．通信プロセスに pCPU を占有させることで，性能が向上することが知られているため [24]，Dom0（通信プロセス）に pCPU を 1 つ占有させる．また，完全仮想化よりも準仮想化のほうが性能が高いことも知られているため [12]，準仮想化を使用する．

本評価で用いるクローラは C 言語で実装されており，実用化されたソフトウェアである [2], [25]．クローラは，各監視点の計測周期を考慮して計測スケジュールを作成する．計測スケジュールで定められた時刻 t_1 になると，クローラはデータ要求の送信処理を開始する．そして，t_1 と，データ要求の送信を完了するまでの時刻 t_2 の差を計測時刻誤差として計算する．ここでの「データ要求の送信」とは，クローラにとっての送信処理であり，実際には Dom0 への送信処理である．

データ要求には BACnet/WS [21] の getValues リクエストを使う．BACnet/WS は HTTP をベースとする通信プロトコルである．ビルの GW は，Apache HTTP サーバ（バージョン 2.4.18）を用いて模擬する．

5.2 実機を用いた評価

実機を用いた評価を実施した．**表 1** に実機評価のパラメータを示す．クローラ/VM は 10 分間隔で 20 個まで追加した．今回，各クローラが管理する監視点数は 100，200，300 のいずれかとし，各監視点の計測周期はすべて 1 秒とした．つまり各クローラの監視頻度（rps; Request per Second）を 100，200，300 のいずれかとした．100 rps は，ビル数棟分の計測に相当する頻度である．また，各クローラの監視頻度が同じ場合と，異なる場合を評価した．監視頻度が異なる場合として，1) 100 rps のクローラと 300 rps のクローラを交互に追加する場合，2) 200 rps のクローラと 300 rps のクローラを交互に追加する場合，3) 100 rps，200 rps，300 rps のクローラを順番に追加する場合の 3 通りを評価した．各 VM の vCPU は 1 個とした．1 個の vCPU は，VM が 1 個であれば，300 rps の処理に対して十分な計算リソースである．

表 1 実機評価のパラメータ
Table 1 Parameters of evaluation with real machines.

パラメータ	値
クローラ/VM の数	1 から 20
クローラ/VM の追加間隔（分）	10
各クローラの監視頻度（rps）	100，200，300
各 VM の vCPU 数	1
クローラ/VM 群が使用できる物理 CPU の数	7
試行回数	10

表 2 シミュレーションのパラメータ
Table 2 Parameters of simulation.

パラメータ	値
クローラ/VM の数	2 から 20
クローラ/VM が使用できる物理 CPU の数	7
各 VM の vCPU 数	1
計測スケジュール	実機評価と同じ
T_{net}（マイクロ秒）	660.855
通信プロセスのタスクキュー長	256
シミュレーション期間（秒）	30
シミュレーションの単位時間（マイクロ秒）	1

評価では計測時刻誤差の 99.999 パーセンタイル値を計測した．以降，単に計測時刻誤差と記述する場合は，計測時刻誤差の 99.999 パーセンタイル値を意味する．評価で得られた計測時刻誤差は，5.3 節でシミュレーションによる推定値と比較する．

実機評価の際，4.4.1 項で述べた方法に基づき，タスクの処理時間を求めた．T_{net} の値は**表 2** に示す．T_{vm} と T_{cp} は rps や VM 数に応じて変化した．その様子を**図 5** に示す．T_{vm} と T_{cp} は，rps が大きくなると小さくなった．Dom0 は各 VM と，専用のメモリ領域を利用してデータを交換する．rps が大きくなると，1 度のメモリコピーで複数のデータ要求/データ応答のデータが交換されるようになるため，タスクあたりの処理時間は減少する．T_{cp} が VM 数に対して反比例する理由も同様であり，ネットワークに対して送

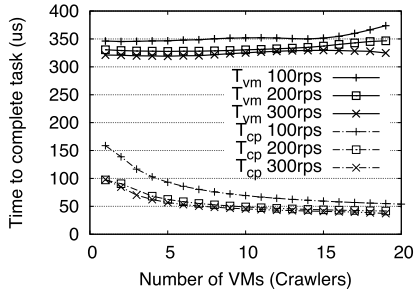


図 5 VM 数，rps とタスクの処理時間の関係
Fig. 5 Time to complete measurement point data task.

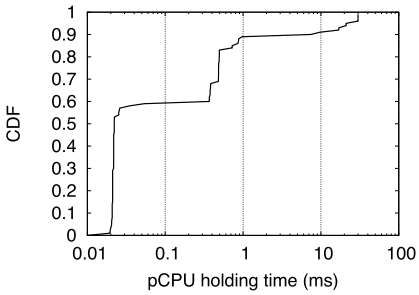


図 6 vCPU が pCPU を解放するまでの時間の CDF
Fig. 6 CDF of time from grabbing pCPU to releasing it.

受信するデータが増加するに従い，Dom0 と NIC の間のデータ交換の効率が上がるためである．

また，4.4.2 項で述べた方法に基づき，タイムスライスの確率密度関数を求めた．vCPU が pCPU を解放するまでの時間は，`xentrace` コマンドを用いて計測した．計測結果の累積分布関数（CDF）を**図 6** に示す．Xen の credit スケジューラの timeslice パラメータを 30 ミリ秒としたため，pCPU 解放までの時間の最大値はほぼ 30 ミリ秒となった．

5.3 実機評価とシミュレーション評価の比較

本節では，実機評価で得られた計測間隔誤差と，シミュレータにより推定した計測時刻誤差を比較した結果について述べる．

シミュレーション評価のパラメータを表 2 に示す．VM の数は 2 個から 20 個までとした．最初に VM を追加する時点では，モニタリング情報が存在しないため，タスクの処理時間（T_{vm} と T_{cp}）を計算できず，よって計測時刻誤差を推定できないためである．また，T_{vm} と T_{cp} は rps と VM 数に応じて変化したため（図 5），最新の値を用いて推定した．たとえば，クローラを 5 個実行する場合の計測時刻誤差を推定する場合は，すでに 4 個のクローラが稼働中であるから，クローラが 4 個のときの T_{vm} と T_{cp} を使用して推定した．通信プロセスのタスクキュー長は，Xen の Dom0 と VM がデータ交換するためのリングバッファのサイズと同じ 256 とした．

また，提案手法に対する比較手法として，近似式による推定手法を用いる．計測時刻誤差は指数的に増加するため，実測値を指数関数 $\exp(a \times \mathrm{VM} + b) + c$ にフィッティングすることで近似式を求め，その近似式を用いて計測時刻誤差を予測する．近似式は，VM を追加するごとに求めなおす．すなわち，実行中の VM 数が N のとき，VM 数が 1 から N までの計測時刻誤差の実測値から近似式を求め，VM 数が $(N+1)$ のときの計測時刻誤差を推測する．この単純な推定手法と比較することで，提案手法の有効性を検証する．

5.3.1 各クローラの監視頻度が等しい場合

図 7 に，各クローラの監視頻度が等しい場合の，計測時刻誤差の実測値と推定値の比較結果をまとめる．図中の `real` は実測値，`proposal` は提案手法，`exp` は近似式による推定値を表す．

計測時刻誤差の実測値は，VM 数に対して指数的に増加している．100 rps の場合は 20 VM でも 5 ミリ秒程度だが（図 7 (a)），300 rps の場合は 18 VM の時点で 100 秒を超えた（図 7 (c)）．そのため，17 VM 以降，グラフがほぼ真上に伸びている．この計測時刻誤差の増大は，Dom0 がクローラ群からの通信要求を処理しきれなくなったために生じている．

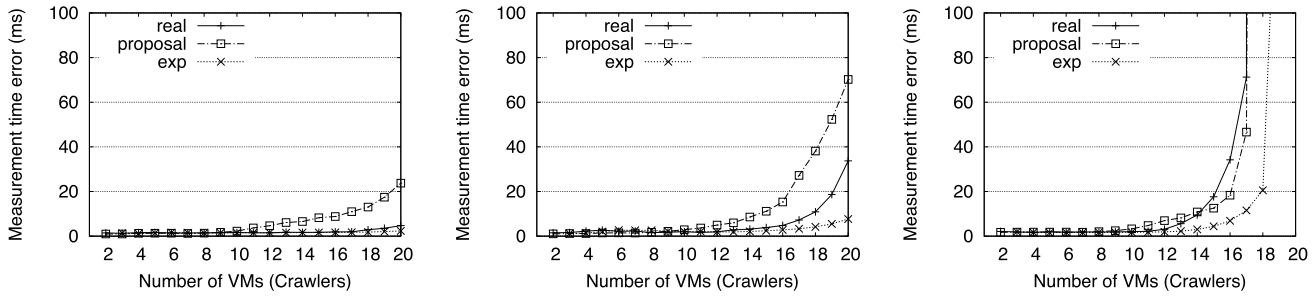
提案手法による推定値は，実測値と同様に，VM 数に対して指数的に増加しているが，増加率を正確に再現できていない．100 rps と 200 rps の場合は，12 VM 以降は実測値よりも大きく推定しているが，300 rps の場合は 15 から 17 VM において実測値よりも小さく推定している．つまり，実測値のほうが推測値よりも，rps の増加に対する計測時刻誤差の増加率が高い．一方，18 VM で計測時刻誤差が増大する傾向は再現できた（図 7）．提案手法は，通信プロセス（Dom0）に VM ごとの有限長のタスクキューを設けたが，このキューがタスクで一杯になることで，Dom0 のボトルネック状態を再現できた．

指数関数による推定値は，実測値よりも低く推定した．特に 300 rps の場合は実測値との差が大きく，計測時刻誤差が増大する VM 数の推定に失敗している．計測時刻誤差の増大は，監視制御機能への影響が大きいことから，最も推定したい現象である．したがって，提案手法は指数関数近似による推定よりも役立つといえる．

5.3.2 各クローラの監視頻度が異なる場合

図 8 に，各クローラの監視頻度が異なる場合の，計測時刻誤差の実測値と推定値の比較結果をまとめる．100 rps と 300 rps のクローラが混在する場合（図 8 (a)）の，各クローラの平均監視頻度は 200 rps である．Dom0 の処理量は各クローラが 200 rps の場合（図 7 (b)）と同等のはずだが，計測時刻誤差の実測値が 20 VM の時点で増大した．100 rps，200 rps，300 rps のクローラが混在する場合も同様の傾向であり，20 VM の時点で増大した（図 8 (c)）．

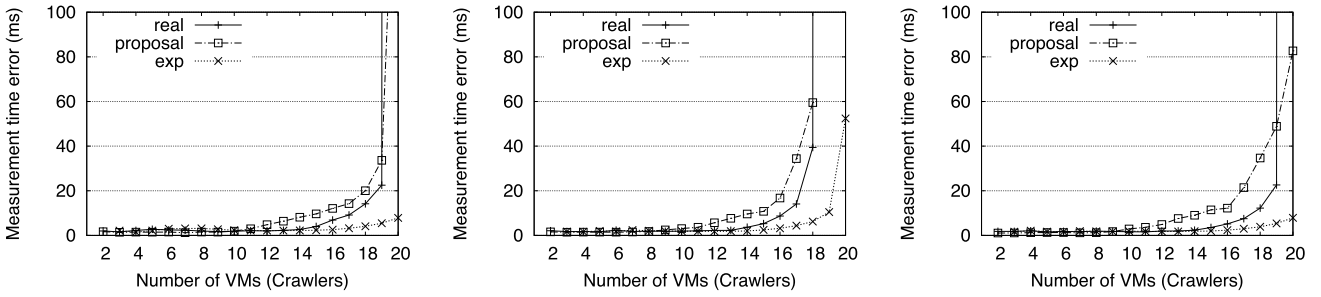
今回，Xen の credit スケジューラをデフォルトの状態で使用した．つまり，VM の負荷の量によらず，各 VM にほ



(a) 全クローラが 100rps　(b) 全クローラが 200rps　(c) 全クローラが 300rps

図 7 計測時刻誤差の 99.999 パーセンタイル値の実測値と推定値の比較（単一の rps）

Fig. 7 Comparison of 99.999th of measurement time errors between actual values and estimated values under constant rps.



(a) 100rps と 300rps のクローラが混在　(b) 200rps と 300rps のクローラが混在　(c) 100rps, 200rps, 300rps のクローラが混在

図 8 計測時刻誤差の 99.999 パーセンタイル値の実測値と推定値の比較（異なる rps）

Fig. 8 Comparison of 99.999th of measurement time errors between actual values and estimated values under different rps.

ぼ均等に CPU リソースが配分される．そのため，200 rps 均一の場合よりも，100 rps と 300 rps が混在している場合のほうが，300 rps のクローラの CPU リソースが不足しやすく，動作が不安定になりやすい．その影響が，20 VM の時点で発生した結果，計測時刻誤差が増大したと考える．

一方で提案手法は，Xen の credit スケジューラの CPU リソースの配分アルゴリズムは考慮していない．これは VM の負荷に応じて CPU リソースが配分される状態であるため，CPU リソース不足によりクローラの動作が遅れるという状況が，実機よりも起こりにくい．そのため図 8 (c) において，提案手法は 20 VM でも計測時刻誤差が増大していない．また，図 8 (a) においても，20 VM の時点で，実測値よりも小さく推定している．ただし，監視頻度が同じ場合（図 7）と同様に，指数関数による推定値と比べると，提案手法による推定値のほうが計測時刻誤差の増加傾向を再現できている．また，図 8 (b) のように，Dom0 がボトルネックになる状況が発生している場合は，計測時刻誤差が増大する VM 数を推定できている．

5.3.3 評価のまとめ

本評価を通じて，提案手法は，指数関数近似による推定よりも計測時刻誤差の増加の傾向を再現できることが分かった．**表 3** に，実測値と推定値の差の平均値をまとめる．全クローラが 100 rps の場合と 200 rps の場合は，近似式による推定手法のほうが差が小さいが，それ以外の場合は提案手法のほうが差が小さい．これは，計測時刻誤差が増大する場合（図 7 (c)，図 8 (a)，図 8 (b)，図 8 (c)）において，提案手法が増大の傾向を再現できているためだと考える．

表 3 実測値と推定値の差の平均値

Table 3 Average difference between actual values and estimated values.

クローラの rps	提案手法	近似式による推定手法
100（図 7 (a)）	4.569	0.394
200（図 7 (b)）	8.600	3.321
300（図 7 (c)）	60,481.341	60,642.515
100，300（図 8 (a)）	700.626	712.007
200，300（図 8 (b)）	5,189.272	5,294.112
100，200，300（図 8 (c)）	714.055	714.717

一部の評価において，提案手法による推定値は実測値よりも小さかった．たとえば，図 7 (c) の 17 VM において，提案手法による推定値は 50 ミリ秒未満だが，実測値は 50 ミリ秒以上である．仮に許容可能な計測時刻誤差が 50 ミリ秒であった場合，提案手法は 17 VM まで追加可能だと判断するが，その結果，計測時刻誤差は許容範囲を超えてしまう．これは監視制御システムにとって問題となる．提案手法の実用性を十分なものとするためには，この「推定値が実測値よりも小さくなる」という現象をなくし，かつ，実測値と推定値の差を数ミリ秒程度まで改善する必要があ

ると考える．この観点から，本評価においては図 8 (a) のみが実用可能な結果であり，そのほかの場合は改善が必要な結果だといえる．

仮想化ソフトウェアの CPU リソースの配分アルゴリズムをシミュレーションのモデルに組み込むことで，推定の精度を高められる可能性がある．仮想化ソフトウェアの挙動を正確にモデル化するほど，推定精度は向上すると考えられるが，それは特定の仮想化ソフトウェアに対する提案手法の依存度を高め，汎用性を低下させることを意味する．複数の仮想化ソフトウェアに共通する CPU リソースの配分アルゴリズムを抽象化して，シミュレーションのモデルに組み込めるとよい．

6. まとめと今後の課題

本論文では，監視制御アプリケーションの 1 つであるクローラを想定し，VM 上で動作するクローラの計測時刻の誤差を推定する手法を提案した．実機を用いた評価結果と，提案手法による推定結果と，指数関数の近似式による推定結果を比較した結果，提案手法は指数関数による推定よりは有用であることが分かったが，その推定精度には改善の余地があることも分かった．

今後は，推定精度の向上のために，シミュレーションのモデルの見直しを検討する．特に，VM ごとの負荷が異なる場合における推定精度を向上させるために，仮想化ソフトウェアの CPU リソースの配分アルゴリズムをモデルに取り込むことを検討する．また，今回の評価で用いた物理マシンとは異なるスペックの物理マシンを用いた評価や，クローラとは別のアプリケーションを用いた評価を行い，提案手法の有効性を検証する．

参考文献

[1] Chaiboonruang, P., Pora, W., Ochiai, H. and Esaki, H.: Small buildings energy management system based on IEEE1888 standard with data compression, *2014 11th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology* (*ECTI-CON*), pp.1–6 (2014).

[2] 伊藤俊夫，米良恵介，金子　雄，松澤茂雄：通信エンドの負荷ピークを低減するためのビル設備情報収集スケジュール作成方法，電子情報通信学会技術研究報告，IN，情報ネットワーク，Vol.111, No.197, pp.77–82 (2011).

[3] Kivity, A., Kamay, Y., Laor, D., Lublin, U. and Liguori, A.: KVM: The Linux Virtual Machine Monitor, *Proc. Linux Symposium*, pp.225–230 (2007).

[4] Barham, P., Dragovic, B., Fraser, K., Hand, S., Harris, T., Ho, A., Neugebauer, R., Pratt, I. and Warfield, A.: Xen and the Art of Virtualization, *SIGOPS Oper. Syst. Rev.*, Vol.37, No.5, pp.164–177 (2003).

[5] Apparao, P., Makineni, S. and Newell, D.: Characterization of network processing overheads in Xen, *1st International Workshop on Virtualization Technology in Distributed Computing, VTDC 2006*, p.2 (2006).

[6] Koh, Y., Knauerhase, R., Brett, P., Bowman, M., Wen, Z. and Pu, C.: An Analysis of Performance Interference Effects in Virtual Environments, *IEEE International Symposium on Performance Analysis of Systems Software, ISPASS 2007*, pp.200–209 (2007).

[7] Givehchi, O., Trsek, H. and Jasperneite, J.: Cloud computing for industrial automation systems – A comprehensive overview, *2013 IEEE 18th Conference on Emerging Technologies Factory Automation* (*ETFA*), pp.1–4 (2013).

[8] Breivold, H., Jansen, A., Sandstrom, K. and Crnkovic, I.: Virtualize for Architecture Sustainability in Industrial Automation, *2013 IEEE 16th International Conference on Computational Science and Engineering* (*CSE*), pp.409–415 (2013).

[9] Breivold, H. and Sandstrom, K.: Virtualize for test environment in industrial automation, *2014 IEEE Emerging Technology and Factory Automation/* (*ETFA*), pp.1–8 (2014).

[10] Felter, W., Ferreira, A., Rajamony, R. and Rubio, J.: An updated performance comparison of virtual machines and Linux containers, *2015 IEEE International Symposium on Performance Analysis of Systems and Software* (*ISPASS*), pp.171–172 (2015).

[11] Li, W. and Kanso, A.: Comparing Containers versus Virtual Machines for Achieving High Availability, *2015 IEEE International Conference on Cloud Engineering* (*IC2E*), pp.353–358 (2015).

[12] Kaneko, Y., Ito, T. and Hara, T.: A measurement study on virtualization overhead for applications of industrial automation systems, *2016 IEEE 21st International Conference on Emerging Technologies and Factory Automation* (*ETFA*), pp.1–8 (2016).

[13] Pu, X., Liu, L., Mei, Y., Sivathanu, S., Koh, Y., Pu, C. and Cao, Y.: Who Is Your Neighbor: Net I/O Performance Interference in Virtualized Clouds, *IEEE Trans. Services Computing*, Vol.6, No.3, pp.314–329 (2013).

[14] Zhou, W., Yang, S., Fang, J., Niu, X. and Song, H.: VMCTune: A Load Balancing Scheme for Virtual Machine Cluster Using Dynamic Resource Allocation, *2010 9th International Conference on Grid and Cooperative Computing* (*GCC*), pp.81–86 (2010).

[15] Tomás, L. and Tordsson, J.: Improving Cloud Infrastructure Utilization Through Overbooking, *Proc. 2013 ACM Cloud and Autonomic Computing Conference, CAC '13*, pp.5:1–5:10 (2013).

[16] Meng, X., Isci, C., Kephart, J., Zhang, L., Bouillet, E. and Pendarakis, D.: Efficient Resource Provisioning in Compute Clouds via VM Multiplexing, *Proc. 7th International Conference on Autonomic Computing, ICAC '10*, pp.11–20 (2010).

[17] Novaković, D., Vasić, N., Novaković, S., Kostić, D. and Bianchini, R.: DeepDive: Transparently Identifying and Managing Performance Interference in Virtualized Environments, *Proc. 2013 USENIX Conference on Annual Technical Conference, USENIX ATC '13*, pp.219–230 (2013).

[18] Zheng, W., Bianchini, R., Janakiraman, G.J., Santos, J.R. and Turner, Y.: JustRunIt: Experiment-based Management of Virtualized Data Centers, *Proc. 2009 Conference on USENIX Annual Technical Conference, USENIX '09*, p.18 (2009).

[19] Bin, J., Girbal, S., Pérez, D.G., Grasset, A. and Merigot, A.: Studying co-running avionic real-time applications on multi-core COTS architectures, *2014 Embedded Real Time Software and Systems* (*ERTS*) (2014).

[20] Zhao, H., Zheng, Q., Zhang, W., Chen, Y. and Huang, Y.: Virtual machine placement based on the VM performance models in cloud, *2015 IEEE 34th International Performance Computing and Communications Conference* (*IPCCC*), pp.1–8 (2015).
[21] ASHRAE: Addendum c to ANSI/ASHRAE Standard 135-2004 – BACnet/WS (2006).
[22] IEEE: IEEE Standard for Ubiquitous Green Community Control Network Protocol, IEEE1888-2011 (2011).
[23] Fraser, K., Hand, S., Neugebauer, R., Pratt, I., Warfield, A. and Williamson, M.: Reconstructing I/O, Technical Report UCAM-CL-TR-596 (2004).
[24] Mahmud, N., Sandstrom, K. and Vulgarakis, A.: Evaluating industrial applicability of virtualization on a distributed multicore platform, *2014 IEEE Emerging Technology and Factory Automation* (*ETFA*), pp.1–8 (2014).
[25] 金子 雄，前川智則，山田孝裕，松澤茂雄：地域エネルギー管理システムの通信ソフトウェアの開発と運用—実証実験により得られた知見，デジタルプラクティス，Vol.5, No.3, pp.213–218 (2014).

金子 雄 （正会員）

2003 年大阪大学工学部電子情報エネルギー工学科卒業．2005 年同大学大学院情報科学研究科博士前期課程修了．同年株式会社東芝入社．社会インフラ・産業システムの研究企画・開発に従事．

伊藤 俊夫

2008 年東京大学工学部電子情報工学科卒業．2010 年同大学大学院工学系研究科電気系工学専攻博士前期課程修了．同年株式会社東芝入社．社会インフラ・産業システムの研究開発に従事．

原 隆浩 （正会員）

1995 年大阪大学工学部情報システム工学科卒業．1997 年同大学大学院工学研究科博士前期課程修了．同年同大学院工学研究科博士後期課程中退後，同大学院工学研究科情報システム工学専攻助手，2002 年同大学院情報科学研究科助手，2004 年同大学院情報科学研究科准教授．2015 年より同大学院情報科学研究科教授となり，現在に至る．工学博士．2000 年電気通信普及財団テレコムシステム技術賞受賞．2003 年本学会研究開発奨励賞受賞．2008 年，2009 年本学会論文賞受賞．2015 年日本学術振興会賞受賞．2017 年大阪科学賞受賞．モバイルコンピューティング，ネットワーク環境におけるデータ管理技術に関する研究に従事．IEEE，ACM，電子情報通信学会，日本データベース学会各会員．