情報処理学会論文誌 Vol.58 No.2 461–470 (Feb. 2017)

ビル設備機器の状態監視における仮想化技術の適用可能性の検討

金子 雄[1,a)] 伊藤 俊夫[1] 原 隆浩[2,b)]

受付日 2016年5月12日, 採録日 2016年11月1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（BMS; Building Management System）には，ネットワークを介して機器の状態を取得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）を利用することで，遠隔 BMS に必要な物理マシン数を減らせるが，仮想化によるオーバヘッドや複数 VM の並列実行によりクローラの性能が低下する．遠隔 BMS は 99.999%という高い稼働率が要求されるため，仮想化がクローラの性能に与える影響を把握しておくことが重要である．本論文では，クローラを VM 上で実行した場合の性能を評価する．この評価により，ミリ秒オーダで CPU 使用率を計測する必要性や，VM 間の CPU 競合が性能に与える影響を明らかにする．

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

A Measurement Study on Applying Hardware Virtualization for Building Facilities Monitoring

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

Received: May 12, 2016, Accepted: November 1, 2016

Abstract: In a remote building management system (BMS) that manages facilities of buildings, there is an application called a *crawler* that collects 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, performance of the crawler running on a VM could be degraded owing to overhead of virtualization and interference among VMs coexisting on the same physical machine. In addition, the crawler needs to meet its performance requirements with high probability, 99.999%, and therefore it is important to clarify the characteristics of the performance degradation. In this paper, we evaluate performance of the crawler running on a VM. From the evaluation, we confirm that 1) CPU usage measurement with millisecond granularity is important for appropriate CPU allocation and 2) CPU contention among VMs is one of the main factors of the performance degradation.

Keywords: monitoring and controlling 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・小規模のビルへの適用が進んでいる．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

[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) hara@ist.osaka-u.ac.jp

り，たとえば以下である．

- **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．
- **警報機能**：設備機器が異常であると判断した場合に，警報をあげる機能．たとえば，ビル管理者向けの監視画面に警告を表示したり，ビル管理者にメールで通知したりする．
- **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，水の流量を調節するバルブを，ある開閉度に維持する場合などがある．

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], [3]．以降，本機能のことを「クローラ」と呼ぶ．上述の監視制御機能は，クローラが計測するデータに基づいて動作するため，クローラは重要な機能である．

遠隔 BMS が管理する各ビルにはオーナ（ビルオーナ）が存在する．ビルオーナごとに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しやすくなる．障害とは，たとえば，あるビルのネットワークや機器の異常により，クローラの動作が遅れたり停止したりすることである．単純には，ビルオーナごとに物理マシンを用意し，各物理マシン上でクローラを実行する構成が考えられる．しかしこの構成は，ビルオーナが保有するビル数が 1 棟でも物理マシンを用意することになり，遠隔 BMS の運用者にとってコストとスペースの増加が問題となる．

この問題を解決しつつ，各クローラの実行環境の独立性を保証するための方法として，計算機の仮想化技術がある [4], [5]．仮想化技術は，仮想的な CPU や NIC などを備えた Virtual Machine（VM）をソフトウェアとして作成する．各 VM は個別の Operating System（OS）を実行できるため，各 VM 上のクローラは，他の VM 上のクローラを意識せずに動作できる．単一の物理マシン上で複数の VM を実行し，各ビルオーナごとのクローラを実行すれば，遠隔 BMS に必要な物理マシン数を減らし，設備投資を削減できる．

一方，VM 上で動作するアプリケーションの性能は，仮想化によるオーバヘッドや複数 VM の並列実行により低下する [6], [7]．クローラを含め，監視制御アプリケーションは高い信頼性が要求されるため [8]，VM の利用がクローラの性能に与える影響を把握することは重要である．VM 上の監視制御アプリケーションの性能を評価した研究として文献 [9], [10] があるが，VM に割り当てる CPU などのリソース量が性能に与える影響や，並列実行する VM 数が性能に与える影響は評価していない．

本論文では，VM に割り当てるリソース量や，並列実行する VM 数がクローラの性能に与える影響を評価する．本論文が明らかにする知見を以下にまとめる．

(1) クローラの性能要件を満たすために必要な CPU リソースを把握するためには，VM の CPU 使用率をミリ秒オーダで計測しなければならない．
(2) 物理マシンの CPU リソースに余裕がある状況でも，VM 数が増えると，クローラの性能要件を満たせなくなる場合がある．
(3) CPU スケジューリングの間隔を調節することで，クローラの性能要件を満たしやすくなる．

これらの知見は，監視制御アプリケーションを VM 上で実行する場合に，その性能要件を満たせるかを判断するために有用であると考える．

以降，2 章で遠隔 BMS と仮想化技術の概要を説明する．3 章で評価の内容を説明し，4 章では，監視制御アプリケーションを実行する VM の管理における課題を述べる．5 章で関連研究について述べ，最後に 6 章でまとめる．

2. 遠隔 BMS と仮想化技術

2.1 クローラによる監視処理

図 1 は，遠隔 BMS とビル群のシステム構成である．遠隔 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，ビル管理者用の操作端末である 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 [11] や IEEE1888 [12], oBIX [13] などの遠隔監視制御向けプロトコルを使い通信する．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）にデータ要求を送信する．GW と LC は，ビル内の監視制御用プロトコルである BACnet [14]

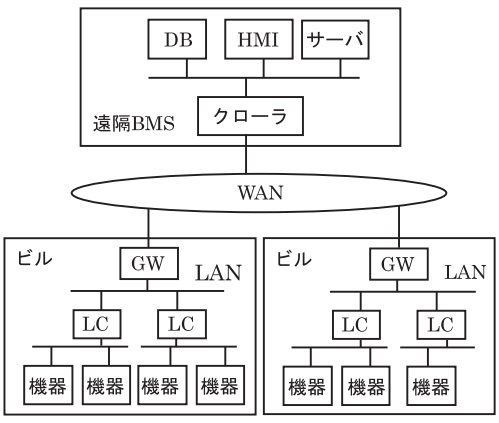


図 1 遠隔 BMS とビル群のシステム構成
Fig. 1 System architecture of a remote BMS and buildings.

などを使い通信する．LC は機器（監視点）から監視点データを取得し，データ応答として GW に返す．GW はデータ応答をクローラに返し，クローラは監視点データを DB に蓄積する．

監視点データを計測した時刻（計測時刻）は，GW が打刻することを想定する．LC や機器は，計測時刻を打刻する能力を持たない場合があるためである．たとえば，BACnet の ReadProperty というサービスにより得られる監視点データには計測時刻が付加されない．したがって，GW が監視点データに打刻する計測時刻と，機器が実際にその状態であった時刻との間には誤差が生じる．しかし，GW と LC は専用の Local Area Network（LAN）により接続されるため，誤差のばらつきはマイクロ秒オーダであり，監視制御機能の品質に影響を与えない．

また本論文では，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自律的に送信するイベント型の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．これらの長所は遠隔 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．

2.2 クローラの性能要件

クローラは，一定間隔で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．以下にその理由を述べる．

- 可視化機能は，監視点データが一定間隔で計測されていれば，ビル管理者にとって見やすいグラフ，すなわち，値が等間隔でプロットされたグラフを作成できる．
- 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．
- 監視点データが一定間隔で計測されないと，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．

許容できる計測間隔の誤差は，監視制御機能の種類や，目指す品質に依存する．たとえば，課金機能や可視化機能は 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．

さらに，クローラを含む監視制御アプリケーションは 99.999%以上の確率で正しく動作することが要求される [8]．99.999%の稼働率は，IEC61508 Safety Integrity Level（SIL）の連続動作モードのレベル 1 として定義される値である．したがって，99.999%以上の確率で，一定間隔で監視点データを計測することをクローラの性能要件とする．

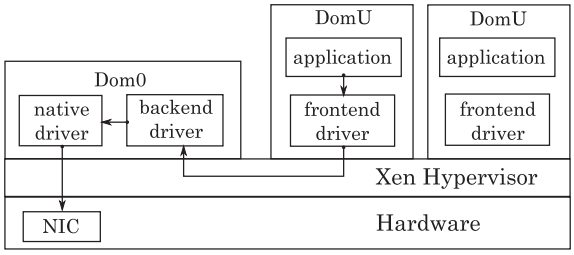


図 2 Xen のアーキテクチャ
Fig. 2 Architecture of Xen Hypervisor.

2.3 仮想化ソフトウェア Xen

本論文では仮想化のソフトウェアとして Xen [5] を使用する．Xen は Hypervisor 型の仮想化ソフトウェアである．**図 2** に Xen のアーキテクチャを示す．

2.3.1 Dom0 と DomU

Xen の特徴は，Dom0 と呼ばれる VM を実行する点である．Dom0 は，DomU と呼ばれる他の VM の起動・停止や，ハードウェアにアクセスするデバイスドライバを実行する権限を持つ．アプリケーションは DomU 上で実行される．たとえばアプリケーションが通信を行う場合は，まず，DomU の frontend driver から Dom0 の backend driver へと通信データが渡される（図 2 の矢印）．そして，Dom0 で NIC のデバイスドライバを実行することで通信を行う．

このように，Xen Hypervisor はデバイスドライバを含まないため，その実装を小さく，単純に保ちやすい．また，デバイスドライバの障害が発生したとしても，その影響が Hypervisor や DomU に波及することを防ぎやすい．たとえば，デバイスドライバの障害を検出して，初期化することが可能である [15]．安定性を重視しているという点で，監視制御アプリケーションに適した設計といえる．

2.3.2 Xen の CPU スケジューラ

本論文の執筆時点で最新の Xen はバージョン 4.6 であり，デフォルトの CPU スケジューラは credit スケジューラである．credit スケジューラは定期的に，各 VM の仮想 CPU（vCPU）に一定量の credit を付与する．そして credit 残量が 0 より大きい vCPU に優先的に，物理 CPU コア（pCPU）をラウンドロビンで割り当てる．各 vCPU は pCPU を使用した時間に応じて，credit を消費する．一方，あまり pCPU を使用しない vCPU の credit 残量は増加していく．credit 残量が所定の閾値まで増加した場合，credit スケジューラはその vCPU の credit の一部を破棄し，他の vCPU に優先して pCPU を割り当てようとする．また，credit スケジューラは timeslice（TS）というパラメータを持ち，基本的に，TS ごとに pCPU の割当てを切り替える．TS のデフォルト値は 30 ミリ秒である．

credit スケジューラは科学計算やバッチ処理のようなアプリケーション向けであり，小さな処理を頻繁に行うアプリケーション（たとえば Web サーバ）向けではない．

pCPU の使用量が小さく，credit が破棄されるためである．そこで，credit2 スケジューラの開発が進められている [16]．credit2 スケジューラは，vCPU の credit 残量が閾値に達しても，破棄せず，閾値にとどめる．また，credit2 スケジューラは credit 残量が多い vCPU に優先して pCPU を割り当てる．したがって，credit スケジューラと比べて，小さな処理を頻繁に行うアプリケーションが pCPU を使用できる機会が増えることが期待できる．

2.3.3 CPU affinity

CPU affinity とは，vCPU と pCPU を関連付ける機能である．Xen では hard affinity と soft affinity を利用できる．hard affinity を使用すると，ある vCPU を実行する pCPU を限定できる．これにより，ある VM に特定の pCPU を占有させられる．soft affinity を使用すると，ある vCPU が，特定の pCPU で実行されやすくなる．これにより，pCPU 間の負荷を緩く調整できる．

3. 仮想化されたクローラの性能評価

3.1 評価の目的

遠隔 BMS の運用者は，単一の物理マシン上で，できるだけ多くのクローラ/VM を実行することで，ビル群を管理するために必要な物理マシン数を減らし，設備投資を抑えたい．ただし，各クローラは 2.2 節で述べた性能要件を満たす必要がある．各物理マシンの性能や各クローラの処理負荷は同一とは限らないため，運用者が各クローラをどの物理マシンに配置するかを考えることは手間である．そのため，各クローラの性能要件を満たしつつ，物理マシンのリソース利用率を最大化してくれる VM 管理手法があることが望ましい．このような VM 管理手法は，Web サーバや DB サーバを対象としてすでに研究されている [17]．しかし，クローラのような監視制御アプリケーションを対象としたものは存在しない．本評価では，クローラを対象とした VM 管理手法の実現に向けて，クローラの性能に影響を与える要因を明確にすることを目的とする．

3.2 評価の環境

図 3 に，仮想化せずにクローラを実行する評価環境（native 環境）と，VM 上でクローラを実行する評価環境（VM 環境）を示す．評価には 2 つの物理マシンを用いる．一方でクローラを動かし，もう一方でビル群の GW を模擬する．2 つの物理マシンは 1000BASE-T で接続する．以降，単に「VM」と述べる場合は DomU のことを指す．

クローラは，各監視点の計測周期を考慮し，データ要求を送る [2]．各監視点の計測周期は 1 秒とする．データ要求には BACnet/WS [11] 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．ビル群の GW は，Apache HTTP サーバで模擬する．Apache HTTP サーバで，データ要求を受信した時刻を記録し，これを計測時刻とする．クローラと GW の通信には，ビルごとに片道 10 ミリ秒の遅延をエミュレーションする．10 ミリ秒という値は，神奈川県にある我々の研究所と，Amazon EC2 の東京リージョンとの間の通信遅延の平均値と同等の値である．2 つの物理マシン間の時刻同期には Network Time Protocol（NTP）を使用する．

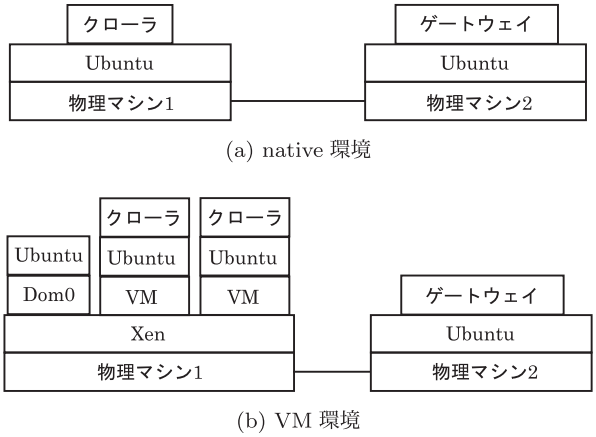


図 3 評価環境
Fig. 3 Experimental environments.

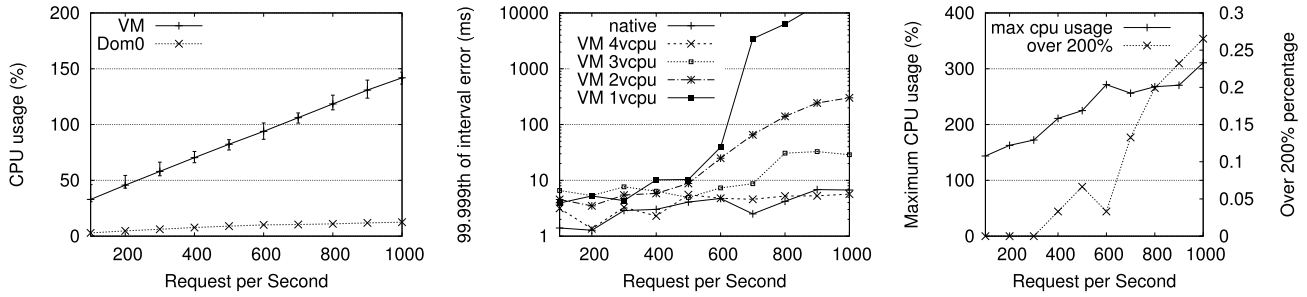
仮想化用の物理マシンは Intel(R) Xeon(R) 1.80 GHz CPU（8 コア），32 GB メモリを備える．仮想メモリアドレスと物理メモリアドレスの変換処理を効率化する Extended Page Tables（EPT）機能は有効化する．ビル GW 用の物理マシンは Intel(R) Xeon(R) 2.60 GHz CPU（32 コア），80 GB メモリを備える．物理マシンの OS として Ubuntu 16.04（Linux 4.4.0）を，VM の OS として Ubuntu 14.04（Linux 3.13.0）を使用する．クローラは C 言語で実装する．DB サーバの性能に起因するクローラの処理タイミングへの影響を除外するため，本評価で用いるクローラは，DB への監視点データの蓄積は行わず，監視点データの計測処理のみを行う．Xen のバージョンは 4.6.0 とする．

3.3 予備実験の結果

本論文で考察すべき対象を絞り込むために実施した予備実験の内容と結果をまとめる．

Xen は仮想化の方法として完全仮想化と準仮想化をサポートしているため，両者におけるクローラの計測間隔誤差を比較した．準仮想化の誤差のほうが数十ミリ秒ほど小さかったため，以降は準仮想化を使用する．準仮想化の性能が優位になることは，従来の研究でも示されている [18], [19]．

また，Xen 4.6 では実験段階のリアルタイム CPU スケジューラである Real-Time Deferrable Server（RTDS）スケジューラを試したが，credit スケジューラや credit2 スケジューラと比べて，計測間隔誤差が数十ミリ秒から数百ミリ秒ほど大きくなった．したがって，以降は credit スケジューラと credit2 スケジューラを使用する．



(a) 平均 CPU 使用率（1 秒計測） (b) native 環境と VM 環境の性能比較 (c) 最大 CPU 使用率（100 ミリ秒計測）

図 4 仮想化されたクローラの CPU 使用率と，vCPU 数と計測間隔誤差の関係
Fig. 4 CPU usage of a virtualized crawler, and the number of vCPUs vs. interval errors.

また，CPU の L3 キャッシュが効きやすくなるように soft affinity を設定したが，計測間隔誤差に変化はなかったため，以降は soft affinity を使用しない．

クローラを実行する VM のリソース消費の傾向を確認したところ，CPU バウンドであったため，以降は CPU リソースに着目して考察する．この評価では，クローラ/VM は 1 個とし，VM の vCPU は 4 個，Dom0 の vCPU は 8 個とした．CPU スケジューラは credit スケジューラとし，CPU affinity は設定せずに評価した．`xentop` コマンドを使用し，VM の CPU 使用率を 1 秒ごとに計測した．

図 4 (a) に計測した CPU 使用率を示す．図中のエラーバーは，計測期間（1 分）における CPU 使用率の最大値と最小値である．図に示すように，クローラを実行する VM の CPU 使用率はデータ要求の送信頻度（rps; Request per Second）に比例することや，CPU 使用率の時間に対する変動は 20%程度であることを確認した．また，クローラ停止時の VM の CPU 使用率は約 0.1%であることを確認した．

3.4 評価の結果

本節では，まず仮想化によるオーバヘッドを評価し，次に複数 VM の並列実行が性能に与える影響を評価する．そして，credit スケジューラと credit2 スケジューラを比較し，最後に CPU affinity が性能に与える影響を評価する．

3.4.1 仮想化によるオーバヘッド

仮想化がクローラの性能に与える影響を評価する．クローラ/VM は 1 個とし，VM の vCPU は 1 個から 4 個とする．vCPU が少ないほどクローラの性能は低下すると予想できるが，その低下の度合いを把握するために，vCPU の数を変化させる．また，Dom0 が通信処理のボトルネックとならないよう，Dom0 の vCPU は 8 個とする．CPU スケジューラはデフォルトの credit スケジューラとする．

クローラの rps は 100 から 1,000 まで，100 きざみとする．中小規模ビルが備える監視点数は数十点から数百点であり，かつ，BACnet/WS の getValues リクエストは，1 リクエストで N 個の監視点データを取得できる[*1]．したがって 1,000 rps は数棟から数百棟のビルの監視に相当し，評価の設定として妥当だと考える．クローラは 99.999%以上の確率で正しくデータを計測する必要があるため，計測間隔誤差の 99.999 パーセンタイル値を評価軸とする．99.999 パーセンタイル値を計算するために，評価期間は 20 分とし，10 万回以上の通信処理をクローラに行わせる．

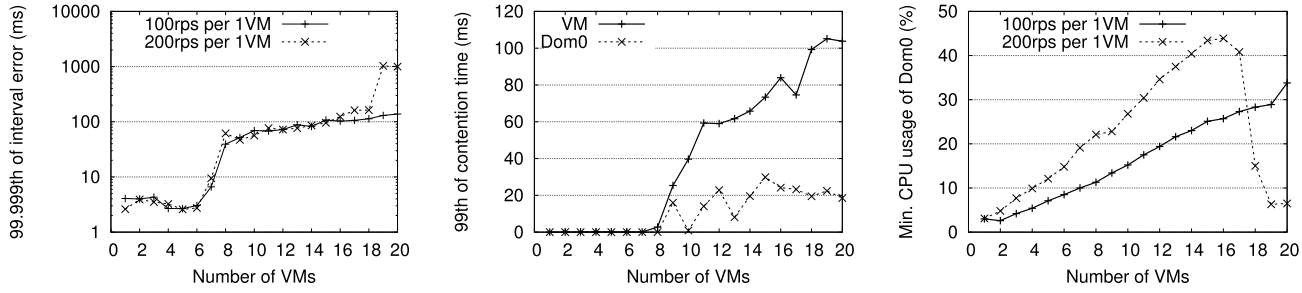
図 4 (b) に評価結果を示す．縦軸は計測間隔誤差の対数軸である．いずれの場合も，rps の増加にともない計測間隔誤差が増加する傾向にある．単位時間あたりの通信回数が増えることで，計画どおりのタイミングで通信を行える確率が減るためである．また，vCPU が 4 個の場合の VM 環境における性能と，native 環境における性能に大差はないことから，本評価で用いたクローラにとって 4 個の vCPU は十分な CPU リソースであるといえる．一方，vCPU が 1 個の場合は，700 rps の時点で誤差が増大した．図 4 (a) から，700 rps の時点で VM の CPU 使用率は 100%を超えることが分かる．つまり，1 個の vCPU では 700 rps の通信処理を行いきれず，誤差が増大した．

vCPU が 2 個の場合は，1,000 rps の時点で誤差が約 300 ミリ秒となった．図 4 (a) によれば，1,000 rps の場合の平均 CPU 使用率は約 140%であり，2 個の vCPU，すなわち 200%の CPU リソースは十分なように思える．しかし図 4 (a) は，`xentop` コマンドを用いて 1 秒間隔で計測した平均値であり，1 秒よりも短い期間において 200%以上の CPU を消費している可能性がある．これを明らかにするため，VM の CPU 使用率をミリ秒間隔で計測するツールを実装した[*2]．本ツールは Xen の `xc_domain_get_cpu_usage` 関数を利用して，VM の CPU 使用率を計算する．

実装したツールで，VM の CPU 使用率を 100 ミリ秒間隔で計測した．VM が真に必要とする CPU リソースを把握するため，十分な CPU リソース，すなわち 4 個の vCPU を VM に割り当てた．結果を図 4 (c) に示す．実線は計測期間中の最大 CPU 使用率を，点線は CPU 使用率が 200%を超えた時間の割合を示している．400 rps の時点で最大 CPU 使用率は 200%を超えている．つまり，100 ミリ秒単位で

[*1] IEEE1888 [12] も同様の機能を備える．

[*2] `xentop` コマンドは，ミリ秒間隔で CPU 使用率を計測できない．



(a) VM 数と計測間隔誤差の関係　(b) VM 数と CPU 競合時間の関係　(c) Dom0 の CPU 使用率の最小値

図 5 複数の VM を並列実行することによる計測間隔誤差への影響

Fig. 5 Impacts of concurrent VMs execution on measurement interval errors.

見ると，200%以上の CPU リソースが必要な時間帯が生じている．その時間帯において，vCPU が 2 個では，CPU リソースが足りないため，クローラの処理が遅れ，計測間隔誤差が増加する．CPU リソースが足りない時間帯の割合は，1,000 rps の場合でも 1%以下だが，計測間隔誤差の 99.999 パーセンタイル値は増加する．

本項の評価により，十分な CPU リソースを VM に割り当てれば，native 環境と同等の性能を達成できることを確認できた．一方，VM に割り当てる CPU リソースを，1 秒間隔で計測した CPU 使用率に基づいて決定すると，計測間隔誤差が数百ミリ秒まで増加する場合があることが分かった．これはフィードバック制御機能の品質に影響を与えうる誤差の大きさである．許容される計測間隔の誤差と同等の間隔で CPU 使用率を計測し，その結果に基づいて割り当てる CPU リソース量を決定すべきである．

3.4.2 VM 数が計測間隔誤差に与える影響

本項では，複数の VM を実行する場合の性能を評価する．多くの VM を実行して評価するため，VM あたりの処理量は小さく設定する．ここでは，各クローラが数棟のビルを監視する状況を想定し，100 rps を設定する場合と，200 rps を設定する場合とを評価する．各 VM には 1 個の vCPU を割り当てる．CPU スケジューラは credit スケジューラとする．その他の設定は 3.4.1 項と同じとする．

図 5 (a) に VM 数と計測間隔誤差の関係を示す．どちらの場合も，VM 数が 8 個の時点で計測間隔誤差が増大している．各 VM の処理量は 100 rps または 200 rps であるから，図 4 (a) によれば，各 VM が必要とする CPU リソースは 50%以下である．本評価に用いた物理マシン 1 は 8 コア，すなわち 800%の CPU リソースを備えるから，8 個の VM に対しては十分だと思える．

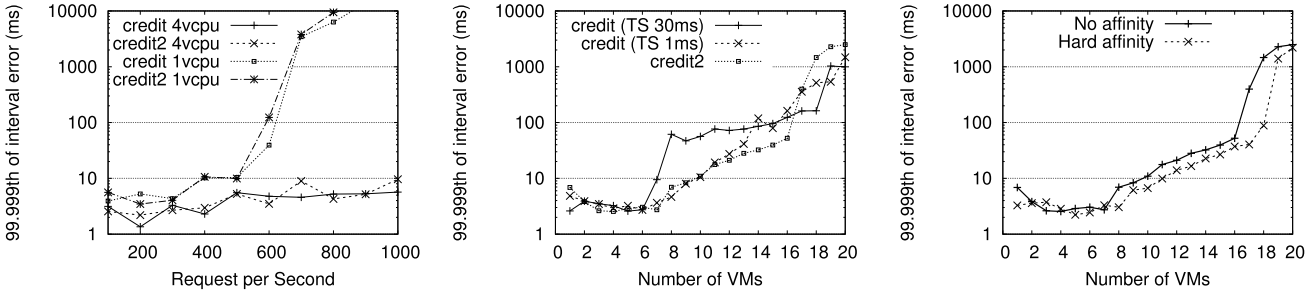
この結果の原因を明確にするために，`xentrace` と `xenalyze` を使用して，各 VM の CPU 競合状態を分析した．CPU 競合状態とは，VM の vCPU が pCPU を使用できずに待機している状態である．分析時の rps は 100 とした．分析の結果を図 5 (b) に示す．縦軸は CPU 競合状態が継続した時間（CPU 競合時間）の 99 パーセンタイル値である．図から，VM 数が 8 個の時点から CPU 競合時間が増加していることが分かる．Xen には VM（DomU）以外に Dom0 が存在する．評価で使用した物理マシン 1 は 8 コアであるため，Dom0 と 8 個の VM のすべてを同時に実行できない．つまり，CPU 競合時間が生じる．CPU 競合状態の間，VM は動作できないため，計測間隔誤差は増加する．

VM の CPU 競合時間が増加し続ける一方で，Dom0 の CPU 競合時間は約 30 ミリ秒以下にとどまっている．本項の評価では，credit スケジューラの timeslice（TS）の値はデフォルトの 30 ミリ秒としたため，Dom0 や VM は pCPU を最大で 30 ミリ秒間占有する[*3]．また，本項の評価では Dom0 に 8 個の vCPU を割り当てたため，Dom0 は VM と比べて pCPU を取得しやすい．したがって，連続する TS において Dom0 が pCPU を取得できないという状況は発生しにくく，Dom0 の CPU 競合時間は TS よりも長くならなかったのだと考える．

VM あたり 200 rps の場合は，VM 数が 19 個の時点で再び誤差が増大し，1 秒を超えた（図 5 (a)）．VM 数が 19 個のとき，全 VM と Dom0 の CPU 使用率の平均値の合計は 722%であった．平均で約 80%の余裕があるにもかかわらず，計測間隔誤差が増大した理由は，Dom0 が CPU を使用できない時間帯が増えたためである．図 5 (c) は，評価期間中の，Dom0 の CPU 使用率の最小値である．この CPU 使用率は `xentop` を用いて 1 秒間隔で計測した．100 rps の場合，Dom0 の CPU 使用率はつねに増加傾向を示している．すべての通信処理は Dom0 を経由して行われるため，想定どおりの結果である．一方，200 rps の場合は，VM 数が 17 個の時点で減少に転じ，VM 数が 19 個と 20 個の時点で約 6%となった．この結果は，VM 群の処理の総量が増加すると，Dom0 が CPU をほぼ利用できない時間帯が発生し，その影響で通信処理のタイミングが乱れ，計測間隔誤差が増大することを示している．

本項の評価結果から，複数の VM を実行する環境では，

[*3] 実行するべき処理がない場合は，30 ミリ秒経過する前に，他の VM に pCPU を引き渡す可能性はある．



(a) 単一 VM における性能比較　(b) 複数 VM における性能比較　(c) CPU hard affinity の有無の比較

図 6 CPU スケジューラと CPU affinity が計測間隔誤差に与える影響

Fig. 6 An impact of CPU schedulers and CPU affinity on measurement interval errors.

計測間隔誤差が増大する箇所が 2 点存在することが分かった．1 点は，VM 群の処理に必要な CPU リソースの総量が増え，Dom0 に十分な CPU リソースを割り当てられなくなったときである．Dom0 はすべての通信処理に関与するため，計測間隔誤差が増大する．もう 1 点は，VM 群により使用される vCPU の数が，pCPU の数を超えたときである．この時点から CPU 競合が発生し，計測間隔誤差が増大する．CPU 競合時間は数十ミリ秒に及ぶため，監視制御機能の品質に影響を与えうる．複数の VM を実行する場合は，各 VM の処理量が小さい場合でも，CPU 競合時間を推定し，監視制御機能の品質に影響が生じるか判断すべきである．

3.4.3 CPU スケジューラが計測間隔誤差に与える影響

3.4.1 項や 3.4.2 項と同様の評価を，credit2 スケジューラを用いて実施し，credit スケジューラの結果と比較する．**図 6** (a) は，単一 VM を実行した場合の計測間隔誤差の比較である．rps や vCPU 数を変化させて比較したが，大きな差は見られなかった．vCPU 数が 1 個かつ 600 rps の場合は，約 60 ミリ秒の差が生じているが，この差はスケジューラの違いによるものではないと考える．rps が 600 の場合，CPU 使用率はときに 100%を超えるため（図 4 (a)），vCPU が 1 個では計測間隔誤差がばらつきやすく，かつ，誤差の 99.999 パーセンタイル値を比較しているため生じた差であると考える．したがって，CPU リソースに余裕がある状況では，両スケジューラの差はないといえる．

図 6 (b) は，複数 VM を実行した場合の計測間隔誤差の比較である．前項で述べたように，credit スケジューラの TS のデフォルト値は 30 ミリ秒であり，これが CPU 競合時間の増加につながっている．そこで，TS を 1 ミリ秒とした credit スケジューラの性能もあわせて評価した．各 VM の rps は 200 とした．

図から，TS が 1 ミリ秒の場合は，VM 数が 8 個の時点でも，誤差を 10 ミリ秒以下に抑えられることが分かる．これは，想定どおり，TS を短くすることで CPU 競合時間を短縮できたためである．`xenalyze` で CPU 競合時間を分析したところ，全体の傾向は TS が 30 ミリ秒の場合（図 5 (b)）と同じだが，絶対値が減少していることを確認できた．このため，VM 数が 8 個になり，CPU 競合が発生する状況になっても，誤差を抑えられた．

VM 数が 14 個以上になると，TS による誤差の差はなくなった．TS を小さくすると，CPU スケジューリングの実行頻度が増すため，CPU スケジューリングによるオーバヘッドが増える．VM 数が 8 個のときは，物理マシンの CPU リソースに余裕があるため，オーバヘッドが増えても，CPU 競合時間の短縮による効果が大きく，誤差を抑えられた．しかし，VM 数が増えると，CPU スケジューリングに必要な処理量は増加する．各 VM の情報を for ループで走査しながら確認する処理が含まれるためである．情報とは，たとえば，各 VM の vCPU の credit 残量である．つまり TS が小さいほど，VM 数が増えたときの，CPU スケジューリングによるオーバヘッドの増加率が高くなる．その分，CPU を通信処理に使用できる時間が減るため，通信処理が遅れる．このような理由から，VM 数が増えた場合に，TS による誤差の差がなくなったと考える．

credit2 スケジューラの CPU スケジューリングの間隔は，500 マイクロ秒から 2 ミリ秒の間である．そのため，credit2 スケジューラの性能は，TS を 1 ミリ秒に設定した credit スケジューラの性能と似ている．ただ，credit2 スケジューラの場合は，VM 数が 16 個の時点まで，誤差の増大を抑えられた．`xenalyze` で分析したところ，各 VM の CPU 使用時間は，credit2 スケジューラのほうが 10%ほど大きかった．2.3.2 項で述べたように，credit2 スケジューラは，使い切れなかった credit を破棄しない．これにより，credit を破棄された影響で CPU を使用できない状況がなくなるため，各 VM の CPU 使用時間が長くなる．つまりクローラはより多くの処理を行えるようになるため，credit スケジューラよりも誤差を抑えられる．

本項の評価結果から，CPU スケジューリングの頻度が高いほど，計測間隔誤差を抑えられることが分かった．credit2 スケジューラは，CPU スケジューリングの頻度が高く，さらに vCPU の credit を破棄しないため，credit スケジューラよりも誤差を抑えられる．ただし，頻繁な CPU

スケジューリングはCPU切替え処理のオーバヘッドを増加させるため，監視制御機能が許容する誤差に応じて，適切にCPUスケジューリングの頻度を設定すべきである．

3.4.4 CPU affinityが計測間隔誤差に与える影響

CPU affinityが性能に与える影響を評価する．hard affinityにより，1個のpCPUをDom0に占有させるよう設定した．また，VM群は残り7個のpCPUを共有するよう設定した．CPUスケジューラはcredit2スケジューラとし，各VMのrpsは200とした．

図6 (c)に結果を示す．hard affinityを使用したほうが誤差が小さいことが分かる．Dom0に専用のpCPUを割り当てることで，DomUとDom0との間でCPUは競合しなくなり，その結果，DomUのCPU競合時間が減少した．そのため，CPU競合が発生するVM数が8個の時点から，誤差が小さくなった．ただしhard affinityを設定した場合でも，VM数が19個の時点で誤差が増大した．`xenalyze`で分析したところ，VMが19個の時点で，DomUのCPU競合の割合が増大していた．VM群の処理の総量が増加し，DomU間のCPU競合が頻繁に生じるようになったため，誤差が増大したと考える．

本項の評価結果から，hard affinityを設定することで誤差を削減できること，また，VM群による処理の総量が一定値を超えると，hard affinityを使用しても誤差の増大は避けられないことを確認できた．Dom0に占有させるべきpCPUの数は，各VMの処理や，各VMの処理により生じるDom0の処理を考慮して決定すべきであるが，その組合せごとに性能評価を行うことは手間である．監視制御機能の性能が最大となるように，自動的にhard affinityを設定する仕組みがあるとよい．

4. VM管理手法の実現に向けた課題

評価実験で得られた知見に基づき，クローラを実行するVMの管理手法の実現に向けた課題を述べる．VM管理手法は，1) VMに割り当てるリソース量の決定と，2) VMを実行する物理マシンの決定を，実施する必要がある．

クローラのCPU使用率は監視点データの計測頻度，すなわちrpsに比例する（図4 (a)）．今回の評価で使用した物理マシンとは異なる物理マシンを使用したとしても，rpsとCPU使用率の関係を導くことは容易であろう．ただし，CPU使用率を計測する場合，その間隔を適切に設定する必要がある．1秒間のCPU使用率の平均値に基づいてvCPUを割り当てると，計測間隔誤差が増加する場合があるためである（3.4.1項）．一方で，細かい間隔（たとえば1ミリ秒）でCPU使用率を計測することは，オーバヘッドを増加させるし，CPU計測の精度を低下させる．許容される計測間隔誤差と同等の間隔でCPU使用率を計測すべきである．

VMに割り当てるリソース量を決定したら，VMを実行する物理マシンを決定する．当然，各物理マシンのリソースの使用状況を把握しておく必要がある．単一物理マシン上に2つ以上のVMを配置する場合は，CPU競合に注意すべきである．VM群が使用するvCPUの数が，pCPUの数を超えた時点で，CPU競合状態が発生する．CPU競合が発生すると，物理マシンのCPUリソースに余裕が存在したとしても，計測間隔誤差は増加する（3.4.2項）．CPUリソースに余裕があれば，CPUスケジューリングの頻度を上げることで，CPU競合による誤差増大を軽減できる（3.4.3項）．Dom0にpCPUを占有させることで，CPU競合による誤差増大を軽減させる方法もある（3.4.4項）．ただし誤差の増加を軽減できるだけで，やはり増加はするため，並列実行するVM数とその処理負荷から，生じうるCPU競合時間を予測したうえで，VMを実行する物理マシンを決定すべきである．また，CPUスケジューリングの頻度やpCPUの占有数は，適切に設定しないと性能が低下する可能性があるため，監視制御機能の性能を最大化できるよう設定すべきである．

5. 関連研究

計算機の仮想化，すなわちVMは，Webサーバや DBサーバなどを中心に幅広く利用されており，近年は監視制御システムに適用するための研究が行われている [20], [21], [22]．VMを利用することで物理マシン数を減らし，障害の波及を制限しつつ，物理マシンにより消費される電力やスペースを削減することが目的である．仮想化を実現するソフトウェアとしてはXen [5] やKVM [4] などがある．2.3.1項で述べたように，安定性を重視した設計であることから，本論文ではXenを使用した．

監視制御アプリケーションをVM上で実行する場合の懸念の1つは性能である．VMはCPUをはじめとするハードウェアをソフトウェアで模擬するため，処理性能が低下する．仮想化がアプリケーションの性能に与える影響は広く研究されており，最大スループットに注目しているものが多い．たとえば，文献 [6] はXenの通信スループットを，文献 [7], [23] は計算処理のスループットを評価している．Webサーバや DBサーバなどのアプリケーションレベルの性能の評価も行われている [24], [25], [26]．

スループットは重要な性能指標であるが，監視制御アプリケーションの場合，処理のスループットよりもタイミングが重要である．文献 [9] は監視制御アプリケーションの1つである Programmable Logic Controller（PLC）を仮想化した場合の処理のタイミングを評価している．文献 [10] は，監視制御アプリを想定してVMの通信遅延を評価している．しかし，どちらの文献も，VMに割り当てるリソース量や，並列実行するVM数が処理タイミングに与える影響を評価していない．並列実行するVM数がアプリケーションの性能に与える影響は文献 [7], [27] などで示されて

いるが，ベンチマークツールを使用した評価であり，監視制御アプリケーションの性能への影響は評価していない．

本論文では，監視制御アプリケーションの具体例としてクローラ [2] を想定し，監視点データの計測間隔の誤差を評価した．そして，VM に割り当てる CPU リソース量や，並列実行する VM 数が計測間隔誤差に与える影響を評価した．さらに，credit2 スケジューラや CPU affinity を使用した評価も実施した．credit2 スケジューラを使用した性能評価は文献 [28], [29] で行われているが，credit2 スケジューラを使用して監視制御アプリケーションの性能を評価した論文は，本論文が初めてであると考える．

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仮想化した場合の性能を評価した．評価の結果，VM に割り当てる CPU リソース量や，VM 数の増加による CPU 競合，CPU スケジューリングの頻度が，クローラの性能に影響を与える要因であることを明らかにした．今後は，本論文で得られた知見に基づき，クローラを実行する VM の管理を自動化する方法を検討する．

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] Debbag, Y. and Yilmaz, E.: Internet based monitoring and control of a wind turbine via PLC, *Smart Grid Congress and Fair* (*ICSG*), *2015 3rd International Istanbul*, pp.1–5 (2015).

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

[5] 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).

[6] 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).

[7] 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).

[8] Grossman, E., Gunther, C., Thubert, P., Wetterwald, P., Raymond, J., Korhonen, J., Kaneko, Y., Das, S., Zha, Y., Varga, B., Farkas, J., Goetz, F. and Schmitt, J.: Deterministic Networking Use Cases, Internet-Draft draft-ietf-detnet-use-cases-10.txt, IETF Secretariat (2016).

[9] Givehchi, O., Imtiaz, J., Trsek, H. and Jasperneite, J.: Control-as-a-service from the cloud: A case study for using virtualized PLCs, *2014 10th IEEE Workshop on Factory Communication Systems* (*WFCS*), pp.1–4 (2014).

[10] 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).

[11] ASHRAE: Addendum c to ANSI/ASHRAE Standard 135-2004 - BACnet/WS (2006).

[12] IEEE: IEEE Standard for Ubiquitous Green Community Control Network Protocol, IEEE1888-2011 (2011).

[13] OASIS: oBIX 1.0, Committee Specification 01 (2006).

[14] ASHRAE: Annex J to ANSI/ASHRAE 135-1995 - BACnet/IP (1999).

[15] Fraser, K., Hand, S., Neugebauer, R., Pratt, I., Warfield, A. and Williamson, M.: Reconstructing I/O, Technical Report UCAM-CL-TR-596 (2004).

[16] Dunlap, G.: Scheduler development update, *Xen Summit Asia* (2009).

[17] Nathuji, R., Kansal, A. and Ghaffarkhah, A.: Q-clouds: Managing Performance Interference Effects for QoS-aware Clouds, *Proc. 5th European Conference on Computer Systems, EuroSys '10*, pp.237–250, ACM (2010).

[18] Deshane, T., Shepherd, Z., Matthews, J.N., Ben-Yehuda, M., Shah, A. and Rao, B.: Quantitative Comparison of Xen and KVM, *Xen Summit* (2008).

[19] Soriga, S. and Barbulescu, M.: A comparison of the performance and scalability of Xen and KVM hypervisors, *2013 RoEduNet International Conference 12th Edition: Networking in Education and Research*, pp.1–6 (2013).

[20] 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).

[21] 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).

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

[23] Babu, S., Hareesh, M., Martin, J., Cherian, S. and Sastri, Y.: System Performance Evaluation of Para Virtualization, Container Virtualization, and Full Virtualization Using Xen, OpenVZ, and XenServer, *2014 4th International Conference on Advances in Computing and Communications* (*ICACC*), pp.247–250 (2014).

[24] Wang, Z., Zhu, X., Padala, P. and Singhal, S.: Capacity and Performance Overhead in Dynamic Resource Allocation to Virtual Containers, *10th IFIP/IEEE International Symposium on Integrated Network Management, 2007, IM '07*, pp.149–158 (2007).

[25] Apparao, P., Iyer, R., Zhang, X., Newell, D. and Adelmeyer, T.: Characterization & Analysis of a Server Consolidation Benchmark, *Proc. 4th ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments, VEE '08*, pp.21–30, New York, NY, USA, ACM (2008).

[26] Mei, Y., Liu, L., Pu, X. and Sivathanu, S.: Performance Measurements and Analysis of Network I/O Applica-

tions in Virtualized Cloud, *2010 IEEE 3rd International Conference on Cloud Computing* (*CLOUD*), pp.59–66 (2010).
[27] Somani, G. and Chaudhary, S.: Application Performance Isolation in Virtualization, *IEEE International Conference on Cloud Computing, 2009*, *CLOUD '09*, pp.41–48 (2009).
[28] Venkataramanan, V.: Optimization of CPU Scheduling in Virtual Machine Environments, Theses University Ottawa (2015).
[29] Hnarakis, R. and Nico, P.: In Perfect Xen, A Performance Study of the Emerging Xen Scheduler, Master's Theses California Polytechnic State University (2013).

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．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．