仮想化が機器状態監視の定刻性に与える影響の一評価

金子 雄[1] 伊藤 俊夫[1] 原 隆浩[2]

概要：ビルや工場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) を利用し，ビルオーナーごとにクローラを実行することで，クローラの障害の影響範囲を限定しつつ，遠隔 BMS に必要な物理マシン数を減らせる．しかし VM の利用は，1) 仮想化のオーバーヘッド，2) 不適切なリソース割り当て，3) 複数 VM の並列実行，などの理由からクローラの性能を低下させる可能性がある．遠隔 BMS は 99.999%という高い稼働率が要求されるため，この性能低下の特性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，様々なリソース量や VM 数において評価する．この評価により，1 秒よりも短い間隔で CPU 使用率を計測する必要性や，複数の VM による CPU 競合が性能低下に与える影響を明らかにする．

1. はじめに

大規模ビルには，空調や照明などの設備機器を効率良く管理するためのビル管理システム (BMS; Building Management System) が導入されている．しかし BMS は高額であり，また，導入のためにはビル内に専用の監視室が必要になるため，中規模・小規模のビルでは導入が進んでいない．そのため，従来の BMS よりも低コストで導入が容易な遠隔 BMS が登場している [1].

遠隔 BMS は、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．そのため，ビル内に監視室を設ける必要がない．遠隔 BMS の管理対象は，空調の運転モードや目標温度，照明のオン・オフ状態，室内の温度や湿度や照度などである．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 の機能は通常の BMS と類似しており，例えば以下である．

- 可視化機能: 監視点データを，ビル管理者にわ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．
- 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．例えば，冷水の流量を調節するバルブの開閉度を，ある状態に維持する場合などがある．
- 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．例えば，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，稼働時間に基づいてテナントに対する課金額を計算する機能．

これらの監視制御機能はいずれも監視点データを必要とする．そのため遠隔 BMS には，ビル群の監視点データを計測し，監視制御機能に提供するための機能が存在する [6][9]．本機能は Supervisory Control And Data Acquisition (SCADA) におけるデータ収集機能に相当する．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ー (ビルオーナー) が存在する．ここで，ビルオーナー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たとしても，他のビルオーナー用のクローラや監視制御機能に影響が及ぶことを避けられる．障害とは，例えば，あるビルのネットワークや機器の異常の影響で，クローラの動作が遅れたり，停止したりすることである．また，監視対象の機器の追加や削除などの設定変更の際に，設定の誤りにより，クローラが停止することである．このような障害による影響の範囲を制限できることが望ましい．

ビルオーナーごとにクローラを稼働する単純な構成は，ビルオーナー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である．この構成は，ビルオーナー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が問題となる．単一の物理マシンかつ単一の Operating System (OS) 上で，複数のクローラを実行する構成も考えられる．この構成はコストとスペースの問題を解決する．しかし，各クローラは，OS が提供するファイルシステム

1 株式会社 東芝 研究開発センター
2 大阪大学 大学院情報科学研究科

などの各種機能を共有することになるため，クローラ間の干渉が生じやすい．例えば，複数のクローラが同じポート番号を使用して通信を試み，通信に失敗する場合が考えられる．

コストとスペースの問題を解決しつつ，各クローラの実行環境の独立性を強く保証するための一手法として，ハードウェアの仮想化技術がある [11][12]．ハードウェア仮想化技術は，物理マシン上に，仮想的な CPU や NIC などを備えた Virtual Machine (VM) を作成する．各 VM は個別の OS を実行できる．すなわち，個別のファイルシステムや通信のアドレス空間などをアプリケーションに提供できる．また，VM のメモリやデバイスの状態などを保存するスナップショット機能を使えば，アプリケーションのバックアップとリストアも容易になる [14]．ライブマイグレーション機能を使えば，アプリケーションを停止することなく，簡単に，別の物理マシンに移動することもできる．

遠隔 BMS の運用者は，個々の物理マシン上に，できるだけ多くのクローラ／ VM を実行したい．そうすることで，遠隔 BMS に必要な物理マシン数を減らし，設備投資や電気料金を削減できる．しかし，VM 上で動作するアプリケーションの性能は，1) 仮想化によるオーバーヘッド，2) 不適切なリソース割り当て，3) 複数 VM の並列実行，により低下する場合があることが知られている [16][17]．遠隔 BMS を含め，監視制御システムは一般的に高い信頼性が要求される [19]．クローラの場合は，正確なタイミングで監視点データを計測し続けなければならない．

そこで本論文では，VM の利用がクローラの性能に与える影響を評価する．特に，VM に与えるリソース量や，単一物理マシン上で同時に実行する VM 数による影響を評価する．本評価により明らかにな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となる CPU リソースを把握するためには，VM の CPU 使用率を，1 秒よりも短い間隔で計測しなければならない．
(2) 個々の VM の CPU 使用率が小さくても，VM の数が物理 CPU の数以上になると，クローラの性能要件を満たせなくなる場合がある．
(3) VM スケジューリングの間隔を短くすることで，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，クローラをはじめとする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する．4 章では，評価の結果に基づいて，監視制御アプリケーションを実行する VM の運用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。

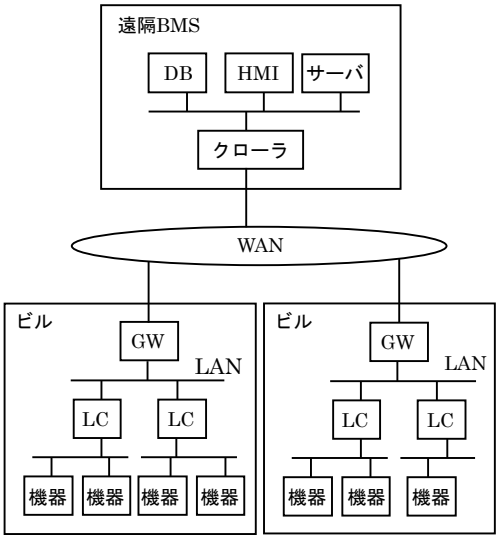

図 1: 遠隔 BMS のシステム構成

2. 遠隔 BMS と仮想化技術

本章では，本論文が想定する遠隔 BMS の概要とクローラの性能要件，仮想化ソフトウェアについて説明する．

2.1 遠隔 BMS による監視処理

図 1 は，遠隔 BMS によるビル群監視制御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．見やすさを考慮して，それぞれ 1 つしか描いていないが，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 の間の通信プロトコルは，BACnet/WS [20] や IEEE1888 [21] や oBIX [22] などの遠隔監視制御向けプロトコルが使用される [24]．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 の間の通信プロトコルは，ビル内監視制御向けのプロトコルである BACnet [25] などが使用される．同様に，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．もしくは，直接，監視制御機能に渡す．

また，本論文ではリクエスト／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能動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．

これによりビル側の機能を簡素化できるし，設定情報の変更も容易である．これらの長所は遠隔 BMS の導入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．

監視点データを計測した時刻 (計測時刻) は，GW または LC が打刻する．本稿では GW が打刻することを想定する．なぜなら，LC が使用する通信プロトコルが，計測時刻を扱えない場合があるためである．この場合は GW が計測時刻を打刻するしかない．例えば，BACnet には ReadProperty というサービスがあるが，これ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．機器が計測時刻を打刻する方法も考えられるが，そのためには機器間で時刻同期が必要になる．他種多用かつ多数の機器間で時刻同期を行うことは，コストの観点から現実的ではない．

GW と LC のどちらが打刻するにせよ，ある監視点データに打刻される計測時刻と，その監視点データが実際に計測された時刻との間には誤差が生じる．一般的に，GW や LC は性能が保証されたハードウェアを用いて実装され，Local Area Network (LAN) により接続される．そのため，データ要求やデータ応答の処理時間のばらつきは，たかだか数ミリ秒である．数ミリ秒のばらつきであれば，監視制御機能の品質に影響はない．

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する必要がある．なぜなら監視制御機能が，一定間隔で計測された監視点データを必要とするからである．以下にその理由を述べる．

- フィードバック制御機能は，監視点データが一定間隔で計測されないと，正しく機器を制御できない．もしくは，制御の品質が低下する．例えば，制御対象の機器の状態が不安定になる．
- 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．
- 可視化機能は，監視点データの推移を折れ線グラフで表示する．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表示できる．

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．例えば，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 秒の誤差は明らかに許容できない．またフィードバック制御の対象機器の特性によっては，100 ミリ秒の誤差を許容できない場合がある．

さらに，遠隔 BMS を含む監視制御システムは 99.999%以上の確率で正しく動作し続けることが要求される [19]．99.999%の稼働率は，IEC61805 で定義される Safety Integrity Level (SIL) の連続動作モードのレベル 1 に相当する稼働率である．したがってクローラは，一定間隔で監視点データを計測するという動作を，99.999%以上の確率で達成する必要がある．

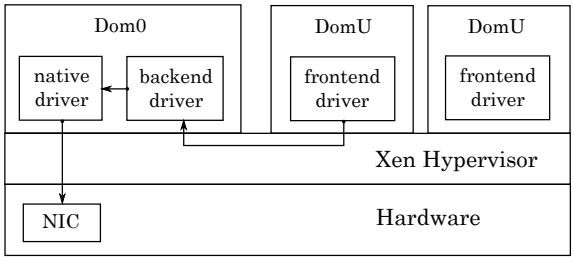

図 2: Xen のアーキテクチャ

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 [12] を使用する．図 2 に Xen のアーキテクチャを示す．Xen は Hypervisor 型の仮想化ソフトウェアであり，物理マシンと VM の間に位置する．

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 [27]．Dom0 は，DomU と呼ばれる他の VM の起動や停止を行うための VM である．また Dom0 は driver domain とも呼ばれ，ハードウェアにアクセスするための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基本的に DomU 上で実行される．例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 backend driver へと通信データが渡される．そして，Dom0 にて NIC のデバイスドライバを実行することで通信を行う．

Dom0 にてドライバを実行するという設計は，Hypervisor の安定性に寄与する．デバイスドライバを含まない分，Hypervisor 自体を小さく，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．例えば，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．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言える．

2.3.2 完全仮想化と準仮想化

Xen は，完全仮想化と準仮想化の両方に対応している．完全仮想化は，CPU や NIC などのハードウェアをソフトウェアでエミュレーションするため，仮想化しない場合と比べて性能が低下する場合が多い．このエミュレーションによるオーバーヘッドは，CPU の仮想化支援機構などを利用することで削減できる．準仮想化は，DomU 上で動作する OS が Hypervisor の存在を意識して動作することで，ハードウェアのエミュレーションによるオーバヘッドを削減する手法である．例えば Linux は Xen の準仮想化に対

応しているため，CPU の仮想化支援がなくてもオーバーヘッドを削減できる．準仮想化は完全仮想化と比べて，計算処理や通信処理の性能が高く，VM 数に対するスケーラビリティも高いことが知られている [29][30].

2.3.3 Xen の CPU スケジューラ

Xen は複数の CPU スケジューラを備えている．本論文の執筆時点で最新の Xen はバージョン 4.6 であり，そのデフォルトの 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，各 VM の仮想 CPU (vCPU) に定期的に credit を付与する．そして，credit 残量が 0 以上である vCPU 群に，物理 CPU (pCPU) をラウンドロビンで割当てる．各 vCPU は pCPU を使用した時間に応じて，credit を消費する．credit 残量が 0 未満となった vCPU は，新たに credit が付与されるまで pCPU を使用できない．credit スケジューラは timeslice というパラメータを持ち，基本的に，timeslice ごとに pCPU の割当てを切り替える．timeslice 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ワークロード向けであり，小さな処理を頻繁に行うワークロード (例えば Web サーバ) 向けではない．後者のワークロードの場合，vCPU は付与された credit を消費しきれない場合が多い．結果，vCPU の credit 残量は増加していく．credit 残量が所定の閾値を超えると，credit 残量は破棄 (0 リセット) される．あまり使用されていない vCPU の credit を破棄することで，他の vCPU を優先するためである．credit が破棄された後に処理が発生しても，ラウンドロビンのため，すぐに pCPU を割当てられる保証はない．また，仮に pCPU が割当てられたとしても，一度 credit 残量を破棄されているため，すぐに残量が 0 未満となる可能性が高い．

credit2 スケジューラは，上記の問題を解決するために開発が進められているスケジューラである [32]．credit2 スケジューラは，credit スケジューラと同様に，vCPU に定期的に credit を付与する．ただし，vCPU の credit 残量が上限値に達しても，破棄しない．credit 残量が上限値を超えないだけである．credit2 スケジューラは，credit 残量が多い vCPU に優先して pCPU を割当てる．したがって，credit スケジューラと比べて，credit の消費量が小さいワークロードが pCPU を利用できる機会が増える．

3. 仮想化されたクローラの性能評価

3.1 評価の目的

VM の利用は，様々な理由でクローラの性能に影響を与える．VM は，CPU や NIC をソフトウェアとして実現するため，仮想化を使わない場合と比べて処理性能が低下する [16]．また，同一の物理マシン上で稼働する複数の VM は，互いの性能に影響を及ぼす [17]．VM に割当てるリソースの量も，その性能に影響を与える．

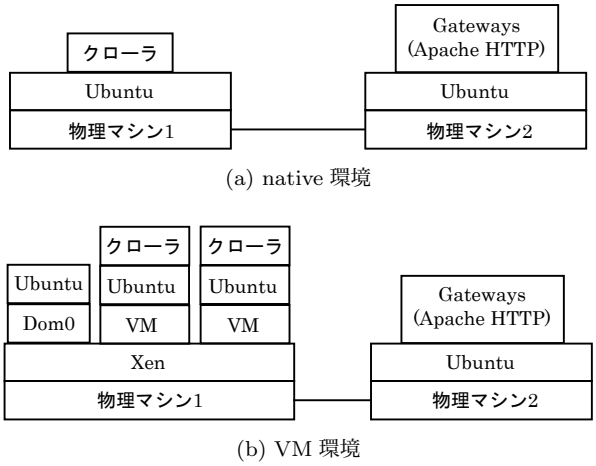

図 3: 評価環境

一方で遠隔 BMS の運用者は，クローラを稼働する物理マシンのリソース (CPU やメモリ) の利用率を最大化したい．つまり，単一の物理マシン上に，できるだけ多くのクローラ／ VM を稼働させることで，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．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するための VM 運用手法があることが望ましい．

このような VM 運用手法は，Web サーバや DB サーバを対象として，多く研究されている [35][36]．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 運用手法の実現に向けて，クローラの性能，すなわち監視点データの計測間隔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

評価の環境について説明する．以降，「VM」と述べる場合は DomU のことを指す．また，仮想化せずにクローラを実行する場合を「native 環境」，VM 上でクローラを実行する場合を「VM 環境」と呼ぶ．

native 環境と VM 環境を図 3 に示す．本評価には 2 つの物理マシンを用いる．1 つはクローラを動かすため，もう 1 つはビルの GW を模擬するためである．2 つの物理マシンは 1000BASE-T で接続する．

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．各監視点の計測周期は全て 1 秒とする．データ要求には BACnet/WS [20] 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 サーバで模擬する．Apache HTTP サーバにて，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．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．

クローラと GW の間の通信には，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．この通信遅延は，東京リージョン上で稼働させた VM に対して，我々の研究所にあるサーバから `ping` コマンドを使用することで計測した [*1]．

仮想化用の物理マシンは Intel(R) Xeon(R) 1.80GHz CPU (8 コア)，32GB メモリを備える．ビル GW 用の物理マシンは Intel(R) Xeon(R) 2.60GHz CPU (32 コア), 80GB メモリを備える．物理マシンの OS として Ubuntu 16.04 (Linux 4.4.0) を使用する．VM の OS として Ubuntu 14.04 (Linux 3.13.0) を使用する．Xen のバージョンは 4.6.0 とし，credit スケジューラと credit2 スケジューラを使用する．また，仮想化の種類は準仮想化を使用する．事前の実験にて，完全仮想化よりも準仮想化のほうが明らかに通信性能が高くなることを確認できたためである．クローラは C 言語で実装する．監視点データの計測間隔の正確さに着目して評価を行うため，評価に用いるクローラは，計測したデータを破棄する [*2]．よって，VM に生じる CPU 負荷のほとんどは，クローラの監視点データ計測の処理により発生する負荷である．また，Apache HTTP サーバはバージョン 2.4.10 を使用する．

3.3 評価の結果

評価の具体的な内容および結果を述べる．まず，デフォルトの CPU スケジューラである credit スケジューラを使用し，本評価で使用するクローラのリソース使用量を把握するための評価を行なう．その後，VM の利用が性能に与える影響，リソース割り当て量が性能に与える影響，複数 VM の同時実行が性能に与える影響の順に評価を行う．最後に，credit スケジューラと credit2 スケジューラの比較を行う．

3.3.1 クローラのリソース使用量の傾向

はじめに，本評価で使用するクローラのリソース使用量の傾向を把握するための評価を行なう．VM 上でクローラを動かし，そのリソース使用量を `xentop` コマンドを用いて 1 秒間隔で計測する．VM は 1 個とし，4 個の仮想 CPU (vCPU) を割り当てる．事前の動作検証により，4 個の vCPU は，クローラを動作させるために十分であることがわかっている．また Dom0 には 8 個の vCPU を割り当てる．全ての通信処理は Dom0 経由で行われるため，Dom0 がボトルネックになることを避けるためである．評価期間は 5 分とする．

[*1] `ping` コマンドにより ICMP エコー要求を 1 秒間隔で 1 週間送信し続け，計測した．

[*2] 実際のクローラは，計測した監視点データを DB に蓄積する処理を行う．

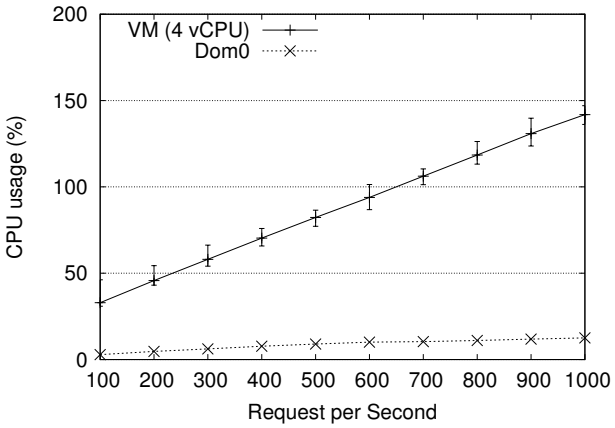

図 4: クローラを稼働する VM の CPU 使用率の平均値

クローラのデータ要求の送信頻度 (rps; Request per Second) は 100rps から 1000rps とする．小規模ビルが備える一般的な監視点数は数十点から数百点であり，仮に全ての監視点を 1 秒間隔で計測したとしても，たかだか数百 rps である．また，BACnet/WS の getValues リクエストは，1 つのリクエストで N 個の監視点データを取得できる [*3]．仮に $N = 10$ とすれば，1000rps は秒間 10000 点の監視に相当する．すなわち，1000rps で動作するクローラは数棟から数百棟のビルを監視可能であり，評価の設定として妥当だと考える．

図 4 は評価期間中に計測した CPU 使用率の平均値である．縦軸は CPU 使用率であり，100%は CPU を 1 つ占有することに相当する．rps の増加に伴い，DomU の CPU 使用率がほぼ線形に増加していることがわかる．また図 2 で示したように，DomU の通信処理は Dom0 を介して行なわれる．したがって，Dom0 の CPU 使用率も rps の増加に伴い増加している．ただしその増加率は VM の増加率と比べると小さい．このように，クローラを実行する VM の CPU 使用量の平均値は rps に比例することや，rps が 700 の時点で 100%を超えることなどを把握できた．

CPU 以外のリソースであるネットワークやメモリ，ディスクの使用量も，rps に比例する傾向を示した．ただし，CPU 以外のリソースの使用量は，絶対値としては小さかった．例えば，1000rps の時点の通信量は約 600Kbps であった．したがって，クローラの処理は CPU バウンドであると言える．以降，CPU リソースに着目して評価および考察する．

3.3.2 仮想化が計測間隔誤差に与える影響

クローラを仮想化することで性能が低下するかを評価する．2.2 節で述べたように，クローラは 99.999%以上の確率で正しく動作する必要があるため，99.999 パーセンタイル値を評価軸とする．VM には 4 個の vCPU を与える．3.3.1 項の評価から，1000rps までであれば，4 個の vCPU

[*3] oBIX や IEEE1888 も同様の機能を備える．

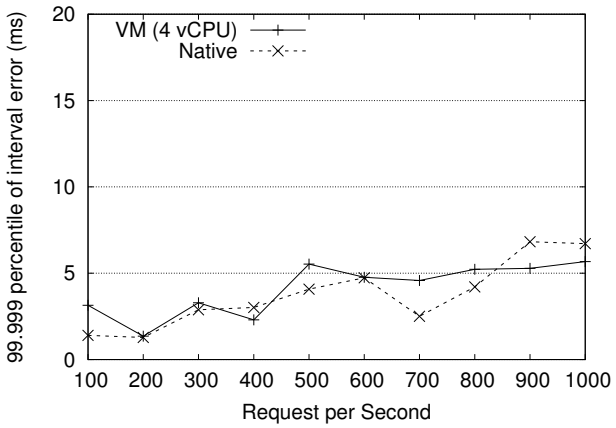

図 5: native 環境と VM 環境の計測間隔誤差の比較

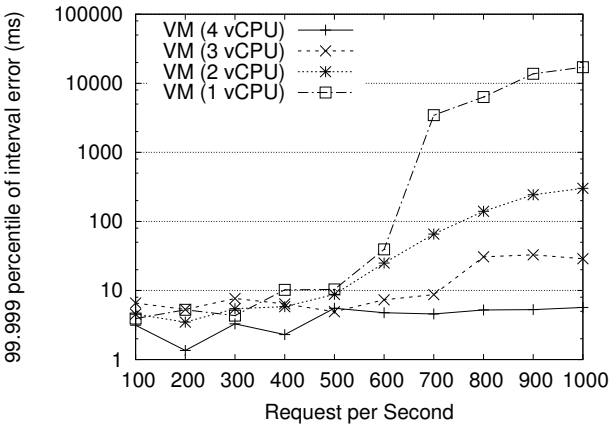

図 6: VM の vCPU 数と計測間隔誤差の関係

(400%) は十分な CPU リソースである．また Dom0 には 8 個の vCPU を割り当てる．評価期間は 20 分とする．20 分とする理由は，計測間隔誤差の 99.999 パーセンタイル値を計算するために，10 万回以上の通信処理をクローラに行わせるためである．

図 5 は native 環境と VM 環境の評価結果の比較である．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，計画通りのタイミングで通信を行なえる確率が減るためである．rps に綺麗に比例した結果になっていないのは，99.999 パーセンタイル値を示しているためである．今回の評価にあたり，OS 上で動作する評価に不要なプロセスは停止した．しかし，例えば NTP などのプロセスは動作しており，それらの動作により計測処理のタイミングがずれ，99.999 パーセンタイル値に影響した可能性がある．このような理由により，図 5 の結果は rps に綺麗に比例した結果になっていないと考える．

native 環境と VM 環境の結果に大きな差は見られなかった．この結果から，VM を利用しても，十分な CPU リソースがあれば，native 環境と大差ない性能を達成できると言える．

3.3.3 vCPU 数が計測間隔誤差に与える影響

3.3.2 項の評価により，十分な CPU リソースがあれば，仮想化による性能低下は見られないことがわかった．そこで本項では，CPU リソースが足りない場合に，クローラの性能がどのように変化するかを評価する．本評価では，VM の vCPU 数を 1 個から 4 個とする．また Dom0 には 8 個の vCPU を割り当てる．評価期間は 20 分とする．

図 6 は評価結果である．縦軸は計測間隔誤差の対数軸である．図 4 の評価から，約 700rps の時点で，VM の CPU 使用率は 100%を超えることがわかっている．したがって，vCPU が 1 個では，700rps の通信処理を行いきれない．そのため，700rps の時点で計測間隔の誤差が増大している．

vCPU が 2 個の場合は，vCPU が 4 個の場合よりも計測間隔誤差が大きくなり，1000rps の時点で約 300 ミリ秒となった．図 4 によれば，1000rps の場合の平均 CPU 使用率は約 140%であるため，2 個の vCPU は十分なはずである．しかし，図 4 の結果は 1 秒間の CPU 使用率の平均値であり，1 秒よりも短い期間においては，一時的に大量の 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 [*4]．本ツールはミリ秒単位の計測間隔を引数に取る．そして，指定された間隔で Xen の `xc_domain_get_cpu_usage` 関数を呼び出すことで各 VM の CPU 消費時間を取得し，CPU 消費時間と計測間隔から CPU 使用率を計算する．

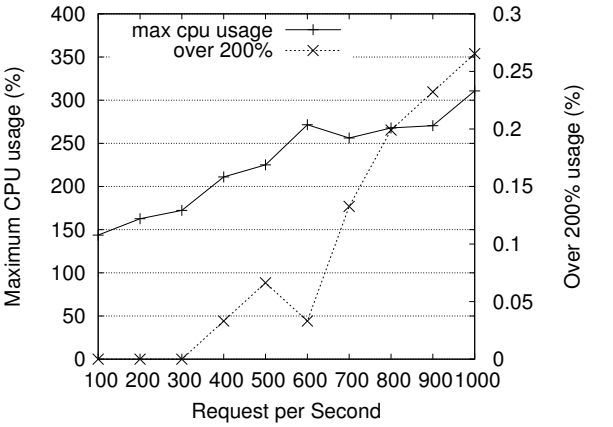

図 7: 100 ミリ秒間隔で計測した CPU 使用率の最大値と，CPU 使用率が 200%を超えていた割合

実装したツールで，4 個の vCPU を割り当てた VM の CPU 使用率を，100 ミリ秒間隔で計測した．4 個の vCPU を割り当てたのは，十分な CPU リソースを割り当てた状況で CPU 使用率を計測することで，真に VM が必要する CPU リソースを把握するためである．その結果を図 7 に示す．実線は計測期間中の最大 CPU 使用率を表している．400rps の時点で，最大 CPU 使用率は 200%を超えている．点線は CPU 使用率が 200%を超えた時間の割合を示して

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

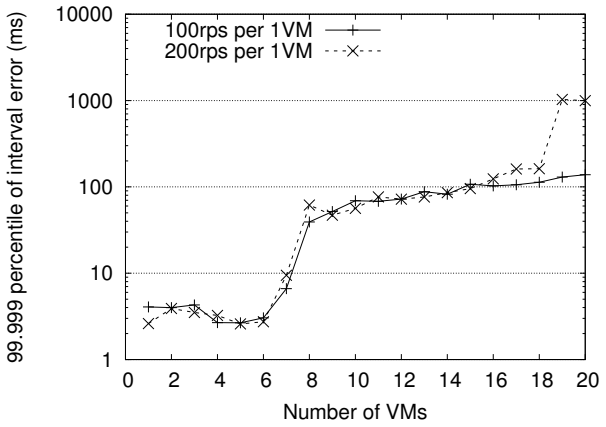

図 8: VM 数と計測間隔誤差の関係

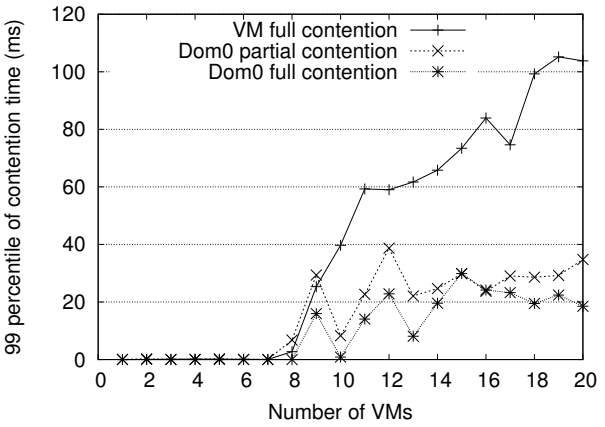

図 9: VM 数と CPU 競合時間の関係

いる．rps の増加に伴い，ある 100 ミリ秒間に CPU 使用率が 200%を超える割合が増加している．これは，100 ミリ秒単位で見ると，2 個の vCPU では CPU リソースが枯渇する時間帯が発生していることを意味する．CPU リソースが枯渇すれば処理が遅れるため，それが計測間隔誤差の増加につながる．一方，図中の最大 CPU 使用率は 400%以下であるから，vCPU が 4 個あれば，100 ミリ秒単位で見た場合でも，十分な CPU リソースであると言える．

本項の評価により，クローラの性能要件を満たすためには，十分な CPU リソースを VM に与える必要があることを確認できた．また，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わかった．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るべきである．

3.3.4 VM 数が計測間隔誤差に与える影響

これまで，物理マシン上に VM を 1 個だけ稼働させた場合の性能を評価してきた．本項では，複数の VM を稼働させた場合の性能を評価する．なるべく多くの VM を稼働させて評価するため，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，100rps を設定する場合と，200rps を設定する場合とを評価する．各 VM には 1vCPU を割当てる．また Dom0 には 8 個の vCPU を割り当てる．評価期間は 20 分とする．

図 8 は評価結果である．横軸は VM の数である．いずれの場合も，VM 数が 8 個のところで計測間隔誤差が増大している．各 VM の処理量は 100rps または 200rps であるから，図 4 の結果によれば，VM が必要とする CPU リソースは 50%以下である．VM 数が 8 個でも $50 \times 8 = 400\%$ である．本評価に用いた物理マシンのコア数は 8 個 (800%) であり，十分な CPU リソースと言えるはずである．

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 が物理 CPU (pCPU) を使用できずに待機している状態である．CPU 競合状態は full contention と partial contention に区別される．full contention とは，VM に割り当てられた全ての vCPU が CPU 競合状態である状況である．partial contention とは，VM に割り当てられた一部の vCPU が CPU 競合状態である状況である．

分析の結果を図 9 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．VM には 1 個の vCPU を割り当てたため，VM には partial contention という状況が生じない．図から，VM 数が 8 個の時点で CPU 競合時間が増加していることがわかる．2.3 節で述べたように，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は 8 コアであるため，Dom0 と 8 個の VM が存在する場合，全てを同時に実行できなくなる．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，計測のタイミングがずれる．

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 20 ミリ秒から 40 ミリ秒の範囲に留まっている．今回，credit スケジューラの timeslice の値はデフォルトの 30 ミリ秒としたため，Dom0 や VM は CPU を最大で 30 ミリ秒間占有する [*5]．一方で，今回の評価では 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 timeslice において Dom0 が pCPU を取得できないという状況は発生しづらく，Dom0 の CPU 競合時間は大きく増加しなかったのだと考える．

VM あたり 200rps 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 8)．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

[*5] 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．

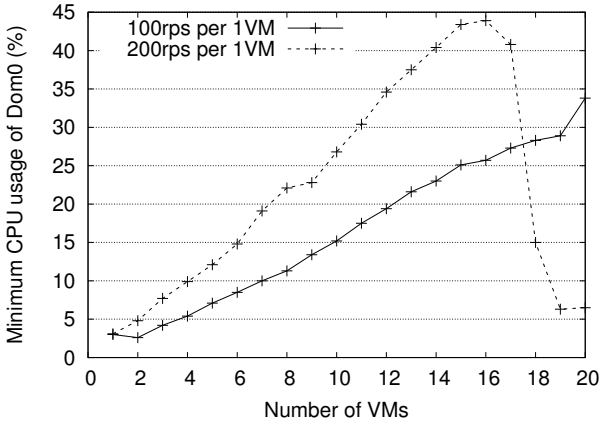

図 10: Dom0 の CPU 使用率の最小値

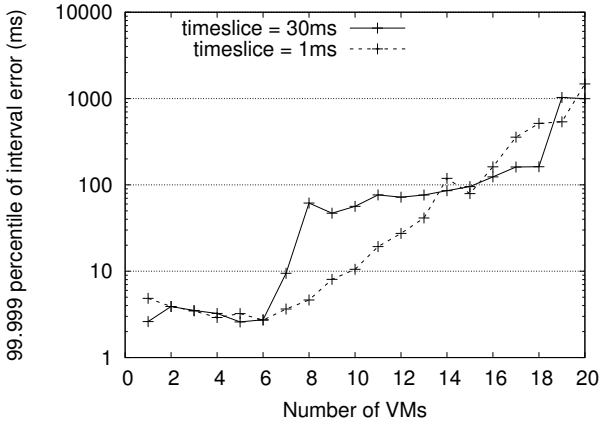

図 11: timeslice が計測間隔誤差に与える影響

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 10 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．100rps の場合，Dom0 の CPU 使用率は常に増加傾向を示している．全ての通信処理は Dom0 を経由して行われるため，想定通りの結果である．一方，200rps 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．本来であれば，45%以上の CPU リソースが必要な状況である．この結果は，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 が共存する環境では，計測間隔誤差が急増する箇所が 2 点存在することがわかった．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 競合状態が発生し，計測間隔誤差が増大する．CPU 競合による誤差増大は数十ミリ秒から数百ミリ秒にわたるため，監視制御機能によっては許容されない場合がありえる．もう 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 は全ての通信処理に関与するため，計測間隔誤差が増大する．物理マシンに新たにクローラ/VM を追加する場合は，この 2 点に留意する必要がある．

3.3.5 timeslice が計測間隔誤差に与える影響

前項で述べたように，credit スケジューラの timeslice のデフォルト値は 30 ミリ秒であり，これが CPU 競合時間の最大値の増加につながっている．本項では，timeslice の値が計測間隔誤差に与える影響を評価する．Xen 4.6 は timeslice をミリ秒単位で設定できるため，1 ミリ秒に設定して評価する．各クローラには 200rps の負荷を与える．その他の評価環境は前項と同じである．

図 11 に評価結果を示す．timeslice が 1 ミリ秒の場合は，

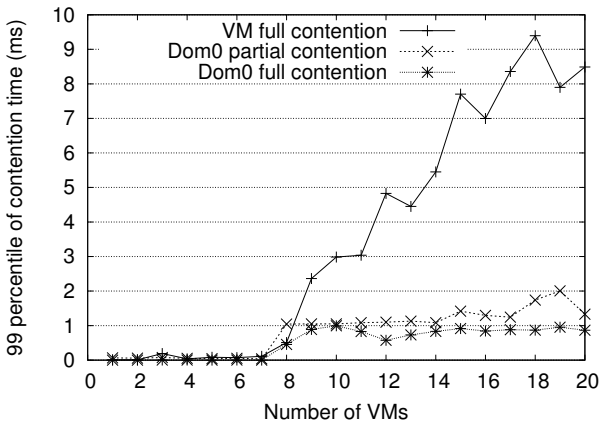

図 12: VM 数と CPU 競合時間の関係 (timeslice=1ms)

VM 数が 8 個の時点でも，誤差を 10 ミリ秒以下に抑えられている．図 12 は timeslice が 1 ミリ秒の場合の CPU 競合時間である．全体の傾向は timeslice が 30 ミリ秒の場合 (図 9) と同じだが，絶対値が減少している．credit スケジューラは，timeslice の間隔で CPU を切り替えようとするため，timeslice の短縮は CPU 競合時間の短縮に寄与する．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，誤差を抑えられる．

一方，VM 数が 14 個以上になると，timeslice による誤差の差はなくなった．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，timeslice を小さくして，CPU 切り替えの頻度を上げることで，各クローラが所望の処理を早く実行できるようになった．しかし，さらに VM 数が増えて，物理マシンの CPU リソースに余裕が無くなると，CPU 切り替えの頻度を上げることによる効果は薄れる．CPU リソースに余裕が無いため，CPU が割当てられるまでの待ち時間は長くなるし，割当てられても，すぐに解放しなければならない．このような理由から，VM 数が増えると timeslice による違いがなくなると考える．本項の評価により，timeslice の値を小さくすることで，クローラの計測間隔誤差を抑えられることを確認できた．

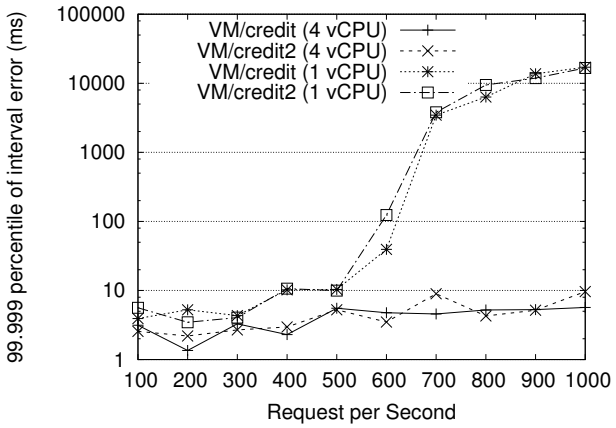

図 13: vCPU 数とスケジューラと計測間隔誤差の関係

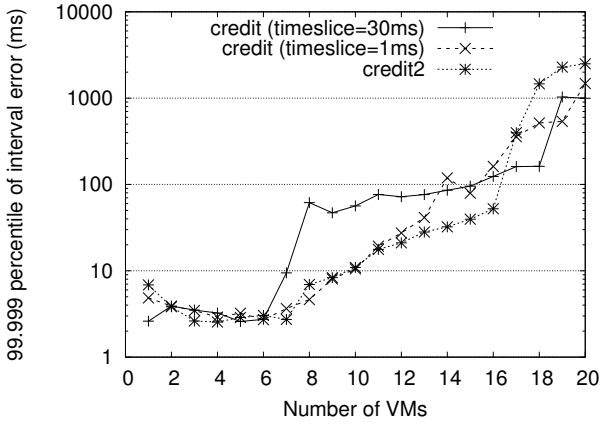

図 14: VM 数とスケジューラと計測間隔誤差の関係

3.3.6 credit スケジューラと credit2 スケジューラの比較

2.3.3 項で述べたように，credit スケジューラよりも credit2 スケジューラのほうが小さい処理を頻繁に行うワークロードに向いている．データ計測という小さい処理を頻繁に行うクローラには，credit2 スケジューラのほうが適している可能性がある．そこで，3.3.3 項や 3.3.4 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 13 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 数を変化させたところ，大きな差は見られなかった．この結果から，CPU リソースに余裕がある状況では，クローラにとって，両スケジューラの差はないと言える．

図 14 は，複数の VM を実行した場合の計測間隔誤差の比較である．Xen 4.6 の credit2 スケジューラの CPU 切り替えの間隔は，500 マイクロ秒から 2 ミリ秒の間である．これは，credit スケジューラにて timeslice を 1 ミリ秒に設定した場合と同等の間隔である．そのため，credit2 スケジューラの性能は，timeslice を 1 ミリ秒に設定した場合に類似している．また，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ている．`xentrace` と `xenalyze` で分析したところ，各 VM が CPU を使用した時間の総量は，credit2 スケジューラのほうが大きかった．2.3 節で述べたように，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．したがって，credit スケジューラよりも credit2 スケジューラのほうがクローラのような監視制御アプリケーションに適していると言える．

4. VM 運用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 の運用手法の実現に向けた課題を述べる．

3.3.2 項で述べたように，必要な CPU リソースを与えれば，仮想化を適用しても，クローラの性能は低下しなかった．したがって，VM が必要とする CPU リソースを正しく見積もれればよい．幸い，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 に比例して決まる (3.3.1 項)．rps はクローラが監視する監視点の数と，各監視点の計測周期から求められる．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．

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 (3.3.3 項)．誤差の増加量は数十ミリ秒から数百ミリ秒だが，監視制御機能の場合はそれが問題になる場合がある．一方で，非常に細かい間隔 (例えば 1 ミリ秒) で CPU 使用率を計測することは，オーバーヘッドを増加させるし，CPU 計測の精度を低下させる．監視制御機能の要求性能に応じて，CPU 使用率の計測間隔を設定する必要がある．

VM が必要とする CPU リソースを見積もれたら，それを割当てられるか，すなわち，物理マシンの CPU リソースに十分な空きが存在するかを判断する必要がある．各 VM が必要とする CPU リソースの合計が，物理マシンの CPU リソースを超えると，計測間隔誤差が急増する (3.3.4 項)．そのような状況を避けるためには，複数の物理マシンの CPU の利用状況を把握し，その情報に基づいて VM の配置先を選ぶ必要がある．

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 群により定常的に使用される 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 リソースに空きが存在したとしても，計測間隔誤差は増加する (3.3.4 項)．CPU リソースに余裕がある状

況であれば，CPU 切り替え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 (3.3.5 項)．Xen であれば timeslice を小さくしたり，credit2 スケジューラを使用することで，CPU 切り替えの頻度を上げることができるため，適切に設定することが望ましい．

5. 関連研究

ハードウェアの仮想化，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いる．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，電気代やスペースを削減することが目的である．

VM の性能や安定性の向上に伴い，近年は，監視制御システムに適用するための研究が行なわれている [38]．監視制御システムのレガシーなハードウェアを仮想化する取り組みや [14]，テスト時に活用しようとする取り組みがある [39]．また，Programmable Logic Controller (PLC) や CAN デバイスなどの具体的な監視制御アプリケーションを仮想化する取り組みもある [40][42]．

ハードウェア仮想化技術と類似する技術として，近年，コンテナ技術が注目されている．コンテナは VM と比べて性能に対するオーバーヘッドが小さいことが知られている [44]．しかし，ライブマイグレーションやスナップショットなどの高可用性に関する機能が充実しているハードウェア仮想化のほうが，監視制御システムの仮想化には適していると考える [46]．

ハードウェア仮想化を実現するソフトウェアとしては Xen [12] や KVM [11] などがある．本論文では Xen を採用する．Xen は Hypervisor 型の仮想化ソフトウェアである．安定性を重視しており，Hypervisor ではなく，Dom0 と呼ばれる専用の VM にてデバイスドライバを実行する設計である [27]．そのため，Hypervisor をコンパクトに保つことができ，デバイスドライバの障害の影響から Hypervisor やゲスト OS を守りやすい．このような Xen の安定性は監視制御システムに適していると考える．

監視制御システムを VM 上で実行する場合の懸念の一つは，性能である．VM は CPU や NIC をソフトウェアで実現するため，仮想化を使わない場合と比べて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．例えば，文献 [16] は Xen の通信スループットを評価している．複数の CPU コアを利用すると，L1 キャッシュにおけるキャッシュミスが増えるため，1CPU コアと比べて CPU コアあたりの性能が低下することを示している．CPU バウンドな処理の性能を評価した研究としては文献 [17] や文献 [48] がある．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に注目した評価も行われている [50][51]．

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 [40] は監視制御アプリケーションの 1 つである 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 [53] 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が処理タイミングに与える影響を評価していない．また，VM 数が物理マシンの CPU コア数を大きく超える状況における評価も実施していない．同一の物理マシンで 2 つの VM が動作するだけで，性能が低下することは複数の研究で示されている [17][55]．

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を想定し，監視点データの計測間隔の誤差を評価軸とした．そして，クローラを VM 上で稼働させた場合の性能を単純に評価するだけでなく，CPU リソースの割り当て量が性能に与える影響を評価した．また，VM 数が物理マシンの CPU コア数を上回る状況における性能も評価した．

本論文では Xen の credit スケジューラと credit2 スケジューラを用いて評価した．文献 [55] や文献 [56] は Xen の credit スケジューラと SEDF (Simple Early Deadline First) スケジューラを比較している．SEDF スケジューラは Xen 4.6 で廃止されている．文献 [47] や文献 [57] は credit スケジューラと credit2 スケジューラを比較している．本論文でも両スケジューラの比較を実施し，credit2 スケジューラのほうが監視制御アプリケーションの処理タイミングを正確に保てることを確認した．

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，監視点データを計測する頻度や，VM に与える CPU リソース，VM の増加による CPU 競合などが，クローラの性能に影響を与える要因であること明らかにした．Xen のデフォルトスケジューラである credit スケジューラよりも，credit2 スケジューラのほうがクローラに適していることも明らかにした．また，クローラを実行する VM の運用を自動化するための課題をまとめた．

今後は，Xen のリアルタイムスケジューラである Real-Time Deferrable Server スケジューラを使用した評価も行なう．また，本論文で得られた知見に基づき，VM の運用を自動化する手法を検討する．

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, *Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON), 2014 11th International Conference on*, pp. 1–6 (2014).

[2] Hong, I., Byun, J. and Park, S.: Cloud Computing-Based Building Energy Management System with ZigBee Sensor Network, *Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS), 2012 Sixth International Conference on*, pp. 547–551 (2012).

[3] Bai, J., Xiao, H., Yang, X. and Zhang, G.: Study on integration technologies of building automation systems based on web services, *Computing, Communication, Control, and Management, 2009. CCCM 2009. ISECS International Colloquium on*, Vol. 4, pp. 262–266 (2009).

[4] Qin, B. and Yan, D.: Remote SCADA System Based on 3G VPN Services for Secondary Pressurization Pump Station, *Intelligent System Design and Engineering Application (ISDEA), 2010 International Conference on*, Vol. 2, pp. 132–135 (2010).

[5] 前川智則，米良恵介，松澤茂雄：インターネットでのポーリング型時系列データ生成方法の一考察 (P2P・一般)，電子情報通信学会技術研究報告. IN, 情報ネットワーク，Vol. 109, No. 449, pp. 109–114 (2010).

[6] 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).

[7] Ninagawa, C., Sato, T. and Kawakita, Y.: Communication performance simulation for object access of BACnet Web Service building facility monitoring systems, *Emerging Technologies and Factory Automation, 2008. ETFA 2008. IEEE International Conference on*, pp. 701–704 (2008).

[8] Parker, P. and Chadwick, S.: Scada approaches to Remote Condition Monitoring, *Railway Condition Monitoring and Non-Destructive Testing (RCM 2011), 5th IET Conference on*, pp. 1–6 (2011).

[9] 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).

[10] Suresh, K., Kirubashankar, R. and Krishnamurthy, K.: Research of Internet based supervisory control and information system, *Recent Trends in Information Technology (ICRTIT), 2011 International Conference on*, pp. 1180–1185 (2011).

[11] Kivity, A., Kamay, Y., Laor, D., Lublin, U. and Liguori, A.: kvm: the Linux Virtual Machine Monitor, *Proceedings of the Linux Symposium*, pp. 225–230 (2007).

[12] 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).

[13] VMware: vSphere ESXi Bare-Metal Hypervisor, VMware (online), available from ⟨http://www.vmware.com/products/esxi-and-esx/overview.html⟩ (accessed 2016-01-26).

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

[15] Bradford, R., Kotsovinos, E., Feldmann, A. and Schiöberg, H.: Live Wide-area Migration of Virtual Machines Including Local Persistent State, *Proceedings of the 3rd International Conference on Virtual Execution Environments*, VEE '07, New York, NY, USA, ACM, pp. 169–179 (2007).

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

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

[18] Yokogawa: FFCS compact control station in centum CS3000 R3, Yokogawa technical report (2004).

[19] Grossman, E., Gunther, C., Thubert, P., Wetterwald, P., Raymond, J., Korhonen, J., Kaneko, Y., Das, S. and Zha, Y.: Deterministic Networking Use Cases, Internet Draft (2015).

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

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

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

[23] Jung, M., Weidinger, J., Kastner, W. and Olivieri, A.: Building Automation and Smart Cities: An Integration Approach Based on a Service-Oriented Architecture, *Advanced Information Networking and Applications Workshops (WAINA), 2013 27th International Conference on*, pp. 1361–1367 (2013).

[24] Schachinger, D., Stampfel, C. and Kastner, W.: Interoperable integration of building automation systems using RESTful BACnet Web services, *Industrial Electronics Society, IECON 2015 - 41st Annual Conference of the IEEE*, pp. 003899–003904 (2015).

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

[26] 西澤茂隆，中村匡秀，井垣宏，松本健一，三浦健次郎：ビル管理システムにおけるサービス指向アーキテクチャの適用: 異種システムの連携と安全性に関する考察 (ユビキタス, 情報配信, ユビキタス時代のネットワークサービス・システム, シームレス通信サービス, 一般)，電子情報通信学会技術研究報告. NS, ネットワークシステム， Vol. 107, No. 261, pp. 3–8 (2007).

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

[28] Camargos, F. L., Girard, G. and des Ligneris, B.: Virtualization of Linux servers, pp. 63–76 (2008).

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

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

[31] Che, J., Yu, Y., Shi, C. and Lin, W.: A Synthetical Performance Evaluation of OpenVZ, Xen and KVM, *Services Computing Conference (APSCC), 2010 IEEE Asia-Pacific*, pp. 587–594 (2010).

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

[33] Nguyen, H., Shen, Z., Gu, X., Subbiah, S. and Wilkes, J.: AGILE: Elastic Distributed Resource Scaling for Infrastructure-as-a-Service, *Proceedings of the 10th International Conference on Autonomic Computing (ICAC 13)*, San Jose, CA, USENIX, pp. 69–82

(2013).

[34] Han, R., Guo, L., Ghanem, M. and Guo, Y.: Lightweight Resource Scaling for Cloud Applications, *Cluster, Cloud and Grid Computing (CCGrid), 2012 12th IEEE/ACM International Symposium on*, pp. 644–651 (2012).

[35] Dhingra, M., Lakshmi, J., Nandy, S., Bhattacharyya, C. and Gopinath, K.: Elastic Resources Framework in IaaS, Preserving Performance SLAs, *Cloud Computing (CLOUD), 2013 IEEE Sixth International Conference on*, pp. 430–437 (2013).

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

[37] Networks, F.: Trends in Enterprise Virtualization Technologies, (online), available from ⟨https://www.f5.com/pdf/reports/enterprise-virtualization.pdf⟩ (2008).

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

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

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

[41] Shahzad, A., Musa, S., Aborujilah, A. and Irfan, M.: A Performance Approach: SCADA System Implementation within Cloud Computing Environment, *Advanced Computer Science Applications and Technologies (ACSAT), 2013 International Conference on*, pp. 274–277 (2013).

[42] Kim, J.-S., Lee, S.-H. and Jin, H.-W.: Fieldbus virtualization for Integrated Modular Avionics, *Emerging Technologies Factory Automation (ETFA), 2011 IEEE 16th Conference on*, pp. 1–4 (2011).

[43] Dua, R., Raja, A. and Kakadia, D.: Virtualization vs Containerization to Support PaaS, *Cloud Engineering (IC2E), 2014 IEEE International Conference on*, pp. 610–614 (2014).

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

[45] Tafa, I., Beqiri, E., Paci, H., Kajo, E. and Xhuvani, A.: The Evaluation of Transfer Time, CPU Consumption and Memory Utilization in XEN-PV, XEN-HVM, OpenVZ, KVM-FV and KVM-PV Hypervisors Using FTP and HTTP Approaches, *Intelligent Networking and Collaborative Systems (INCoS), 2011 Third International Conference on*, pp. 502–507 (2011).

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

[47] Hnarakis, R. and Nico, P.: In Perfect Xen, A Performance Study of the Emerging Xen Scheduler (2013).

[48] 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, *Advances in Computing and Communications (ICACC), 2014 Fourth International Conference on*, pp. 247–250 (2014).

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

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

[51] Mei, Y., Liu, L., Pu, X. and Sivathanu, S.: Performance Measurements and Analysis of Network I/O Applications in Virtualized Cloud, *Cloud Computing (CLOUD), 2010 IEEE 3rd International Conference on*, pp. 59–66 (2010).

[52] Patnaik, D., Krishnakumar, A. S., Krishnan, P., Singh, N. and Yajnik, S.: Performance Implications of Hosting Enterprise Telephony Applications on Virtualized Multi-core Platforms, *Proceedings of the 3rd International Conference on Principles, Systems and Applications of IP Telecommunications*, IPTComm '09, New York, NY, USA, ACM, pp. 8:1–8:11 (2009).

[53] Mahmud, N., Sandstrom, K. and Vulgarakis, A.: Evaluating industrial applicability of virtualization on a distributed multicore platform, *Emerging Technology and Factory Automation (ETFA), 2014 IEEE*, pp. 1–8 (2014).

[54] Huang, Q. and Lee, P. P.: An Experimental Study of Cascading Performance Interference in a Virtualized Environment, *SIGMETRICS Perform. Eval. Rev.*, Vol. 40, No. 4, pp. 43–52 (2013).

[55] Somani, G. and Chaudhary, S.: Application Performance Isolation in Virtualization, *Cloud Computing, 2009. CLOUD '09. IEEE International Conference on*, pp. 41–48 (2009).

[56] Cherkasova, L., Gupta, D. and Vahdat, A.: Comparison of the Three CPU Schedulers in Xen, *SIGMETRICS Perform. Eval. Rev.*, Vol. 35, No. 2, pp. 42–51 (2007).

[57] Venkataramanan, V.: Optimization of CPU Scheduling in Virtual Machine Environments (2015).