計算機システムエミュレーションにおける再現性の評価

福意大智[1] 水本旭洋[2] 西本真介[2] 金田茂[3] 高井峰生[3] 木村啓二[1]

概要： ネットワークの研究では実験のためにネットワークシミュレータが利用されることが多い．しかし，シミュレーションを行うためにはアプリケーションなどの評価対象のモデルをシミュレータに実装する必要があり，評価対象によってはモデル化の労力が大きくなる．また，モデル化による誤差のために実際とは異なる結果が得られる可能性がある．モデル化の労力と誤差を軽減してシミュレーションをより容易にするため，シミュレータと計算機システムエミュレータ QEMU を組み合わせ，アプリケーションを QEMU 仮想マシン上で直接実行することが有効であると考えられるが，仮想マシン上ではアプリケーションの実行時間が実行毎に変動する．そこで，本稿では仮想マシン上で実行時間が変動する要因を調査し，スナップショットの利用によりアプリケーションの実行を再現した場合，実行時間がどの程度変動するのか調査した．要因の調査の結果，QEMU 仮想マシンのタイマと I/O 制御が実行時間の変動に影響していることを確認した．さらに，スナップショットの利用により，アプリケーション実行時間の変動として Intel x86 アーキテクチャをエミュレートした場合では 10 us から 70 us のばらつきが，ARM Cortex-A9 アーキテクチャをエミュレートした場合では 3ms から 124ms のばらつきがそれぞれ生じることを確認した．スナップショットを利用しない場合では，Intel x86 アーキテクチャをエミュレートした場合では 400 us から 20 ms のばらつきが， ARM Cortex-A9 アーキテクチャをエミュレートした場合では 50 us から 18 ms のばらつきが認められた。

An Evaluation of the Repeatability of Full Computer System Emulation

DAICHI FUKUI[1] TERUHIRO MIZUMOTO[2] SHINSUKE NISHIMOTO[2]
SHIGERU KANEDA[3] MINEO TAKAI[3] KEIJI KIMURA[1]

1. 研究概要

ネットワークの研究においてプロトコルやアプリケーションの性能評価を行う際，ネットワークシミュレータによるシミュレーションによって評価が行われることが多い．シミュレーションによる評価では，様々な環境を想定した評価を行うことができ，実機による評価と比べ再現性が高く，再評価も容易である．

しかし，シミュレーションを行うためには評価対象のシステムをモデル化してシミュレータに実装する必要があるため，多種の Android 端末接続を前提とした通信システムなど，評価対象によってはモデル化の労力が大きくなる．また，モデル化による誤差のために実際とは異なる結果が得られる可能性がある．これらの問題は，対象端末全体をエミュレートする計算機システムエミュレータを用いて解決することができると考えられる[3]．本研究が想定するネットワークシミュレータと計算機システムエミュレータを組み合わせたシステムの概略図を図 1 に示す．

ネットワークシミュレーションではネットワークのトポロジやルータの数，端末の数，障害物や輻輳の設定とったパラメータを変更することでネットワークをモデル化する．一方，仮想マシン（VM）は複数台用意して評価に使用するアプリケーションをリアルコードとして直接実行する．

しかし，一般的に計算機システムエミュレータはシステム全体を仮想的に再現しているため，プロセスの割り込みやディスクアクセスといった外乱が任意のタイミングで生じ，アプリケーションの実行状態が再現されないおそれがある．すなわち，同一のパラメータでシミュレーションを行ったとしても，ネットワークアクセスが行われるタイミングなどのイベント発生時刻がシミュレーション毎に変化するため，シミュレーション結果が再現されない可能性がある．

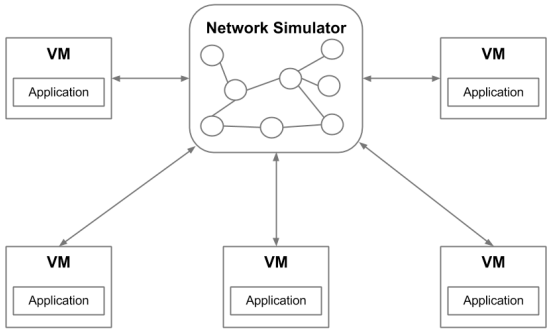

図 1 ネットワークシミュレータと計算機システムエミュレータを組み合わせたシステム

本研究ではアプリケーション実行の再現性を向上させることを目的に，シミュレーションを行う度にスナップショットを用いてアプリケーション実行時のシステム状態を復元する手法を提案する．本稿では，スナップショットにより実行環境を再現することで，どの程度アプリケーションの実行時間が再現されるのか評価を行う．

評価では計算機システムエミュレータとして，広く使われている QEMU[1] を使用する．アプリケーションの評価

1 早稲田大学
2 株式会社スペースタイムエンジニアリング
3 Space Time Engineering，LLC

を行う際の動作環境として，CPU が Intel x86 アーキテクチャの環境と ARM Cortex-A9 アーキテクチャの環境を構築し，これらの環境における実行時間の誤差の評価，およびその要因の検討を行う．

本稿の以降では第 2 節で計算機システムエミュレータ QEMU の概要を示し，第 3 節でスナップショットを用いてアプリケーションの実行時間を再現する手法を提案し，本手法を用いた評価の結果とその考察を第 4 節に示す．第 5 節では関連する研究について議論し，第 6 節では全体のまとめについて述べ，評価結果をもとに今後の課題について議論する．

2. 計算機システムエミュレータ QEMU の概要

QEMU は計算機システムエミュレータとして広く利用されており，ホストのアーキテクチャに依存せず様々なアーキテクチャをエミュレートすることが可能である．また，CPU だけではなくメモリやディスク，NIC などの計算機システム全体の挙動をエミュレートでき，各々のトレースを取得し挙動を解析することが可能である．QEMU にはシステムのエミュレーションの他にも様々な機能が備わっており，本節では，提案手法に深く関連する，(1) QEMU における時刻管理，(2) 仮想マシン上の I/O 制御，(3) 仮想マシンのシステム状態を復元するスナップショットについて記述する．

2.1 QEMU における時刻管理

QEMU は Intel x86 アーキテクチャに備わっている TSC (Time Stamp Counter) や HPET (High Precision Event Timer) といったハードウェアタイマをエミュレートすることができる．QEMU 上のゲスト OS として Linux OS を使用する場合の時刻参照システムを図 2 に示す．Intel x86 アーキテクチャ上では一般に Linux カーネルはこのようなハードウェアタイマを参照することでシステム時刻を設定しており，QEMU の仮想マシン上で動作している Linux カーネルもエミュレートされたタイマを参照することでシステム時刻を設定する．QEMU ではゲスト環境のハードウェアタイマはホスト環境のハードウェアタイマを参照することで時刻を決定している．

QEMU ではハードウェアタイマをエミュレートするために icount という機能が用意されている．icount とは QEMU に実装されている命令数カウンタであり，仮想マシン上の時間で 2^N ナノ秒に 1 回だけ命令を実行することができる．N は仮想マシン起動時のオプションとして，0 以上の任意の値を設定できる．icount を利用する場合，仮想マシンのハードウェアタイマはホストマシンのハードウェアタイマではなく QEMU の icount の値を参照するため，タイマに関してはゲスト環境がホスト環境から影響を受けることはない．

icount を用いる場合はホストのタイマとは無関係にゲストのタイマが進行するため，ゲスト上で実行するアプリケーションの実行時間がホストのものとは大きく異なる可能性がある[2]．しかし，一般にシミュレーションではパラメータを変更するとシミュレーション結果がどのような傾向を示すのかを調査することが目的であるため，ゲスト OS におけるアプリケーションの実行時間がホスト OS におけるアプリケーションの実行時間と必ずしも一致する必要はない．

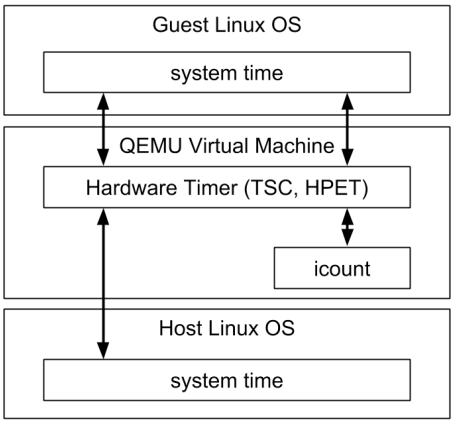

図 2 QEMU における時刻参照システム

2.2 I/O 制御

QEMU ではゲスト環境で発行された I/O を適切に処理するための機能が備わっている．QEMU ではゲストで I/O が発行されると，仮想的なディスクへのアクセスが生じる．QEMU の仮想マシンからアクセスされるディスクは，ホストマシン上にハードディスクイメージファイルとして保存されている．QEMU は，このハードディスクイメージファイルを仮想マシンにおけるハードディスクドライブとしてエミュレートしている．そのため，ゲスト OS で発行された I/O 処理はホスト OS 上のファイルディスクリプタを通してファイルアクセスに変換される．

QEMU では，仮想マシンの CPU と I/O 処理を vcpu スレッドと worker スレッドという，2 つのスレッドにより実装している．これら 2 つのスレッドが協調して動作することで仮想マシン上においてリアルコードのプログラムが実行される．すなわち，worker スレッドはゲスト上で発行された I/O をホストに遷移させて処理する．一方，vcpu スレッドは I/O 処理による待ち時間が発生している間にゲスト上で発行された他の命令を処理する．QEMU は，このような 2 つのスレッド用いて，I/O 処理とゲストコード実行を並行に行うことで全体のパフォーマンス向上を行っている．

2.3 スナップショット機能

QEMU ではある時点における CPU やディスクといったシステムの状態を保存し，後にそのシステム状態を復元するというスナップショット機能が用意されている．本機能を利用することで評価環境を再現し，仮想マシンにおいて常に同じ環境の下でアプリケーションを評価することができる．

QEMU ではゲスト環境でのディスクを読み取り専用にし，ディスクへの I/O を全てバッファに転送することでゲスト環境のディスクが変更されるのを防ぐという機能もスナップショットと呼ばれているが，以降の節でいうスナップショットとはシステム状態を復元するものを指す．

3. 提案手法

一般に計算機システムエミュレータは計算機システム全体をエミュレートしているため，ゲスト環境において割り込みといった外乱が任意のタイミングで発生し，アプリケーション実行時間の再現性に影響を与える．また，ゲストにおけるメモリの状態やディスクの状態，他プロセスも外乱としてアプリケーションの実行に影響する．外乱によりアプリケーション実行時間の再現性が低下する例を図 3 に示す．アプリケーション実行時におけるキャッシュやメモリの状態により，ページフォルトの発生する様子が異なり，アプリケーションの実行時間が変化する．また，アプリケーション実行中において CPU 動作周波数の変化や I/O 処理，割り込み，他プロセスとのスケジューリングによっても実行時間が変化する．

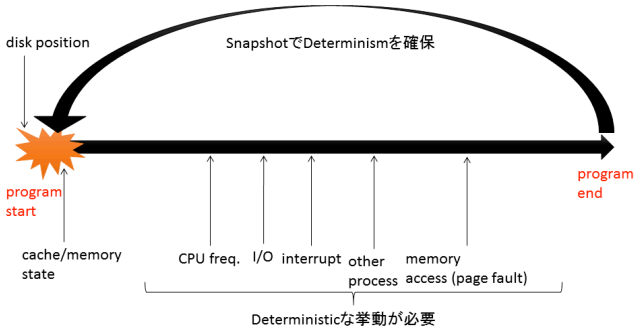

図 3 アプリケーション実行時間の再現性の低下

これらの外乱に個別に対応すると煩雑であるため，本手法ではスナップショットを用いてアプリケーション実行前におけるシステムの状態を復元することで，アプリケーションの実行時間を再現するという手法を提案する．本手法ではスナップショットによりアプリケーションを実行する瞬間におけるシステムの状態，すなわち初期状態を復元する．そしてキーボード割り込みなどの外乱がなければ実行後のスケジューリングや命令の発行が再現され，実行時間も再現されると考えられる．この手法のもとでアプリケーションの実行時間が再現されれば，ネットワークシミュレータにおいてトポロジなどのパラメータを変更した際にシミュレータ上の通信時間が変化した場合，その結果は仮想マシンの外乱ではなくパラメータに変化によるものであると断定できるようになる．次節では，本手法によりアプリケーション実行時間の変動がどの程度抑制できるのか評価を行う．

4. 評価と考察

4.1 評価環境と評価方法

本節では評価のために用いた環境を説明する．QEMU そのものを動作させるホストの環境を表 1 に示し，QEMU により実装された仮想マシンの環境を表 2 と表 3 に示す．仮想マシンは icount を有効にして起動している．評価で使用したアーキテクチャは，サーバや無線通信端末として広く使われている Intel x86 と ARM Cortex-A9 である．QEMU は命令セットをエミュレートするため，ゲスト環境とホスト環境のアーキテクチャが異なっていても動作が可能である．

評価対象のアプリケーションとして，SPEC[6] が提供するベンチマーク群に含まれる art，equake，hmmer を利用した．また，メディア処理の性能評価に利用される MediaBench に含まれる mpeg2encoder を利用した．

本評価では，アプリケーションを実行する直前のシステム状態におけるスナップショットを取得し，そのスナップショットを復元することでアプリケーションの実行時間がどの程度再現されるのか調査した．スナップショットの復元を 50 回程度行い，実行時間の最大値，最小値，分散を求め，実行時間のばらつきを確認する．また，スナップショットを使わずに同じアプリケーションを 50 回程度繰り返して実行した場合のおける実行時間の最大値，最小値，分散も求める．

4.2 評価結果と考察

表 4 より，Intel x86 の仮想マシンではスナップショットを用いた場合，アプリケーションの実行時間に 10 us から 70 us のばらつきが認められる．一方，表 5 では ARM Cortex-A9 の仮想マシンにおいてスナップショットを用いた場合，3 ms から 124 ms のばらつきが認められる．この結果よりスナップショットにより仮想マシンのシステムの状態を復元した上でアプリケーションを実行しても厳密に実行時間は再現されず，ある程度のばらつきが発生することが確認できた．特に，ARM CortexA-9 仮想マシンではばらつきが Intel x86 仮想マシンに比べて大きくなっている．

表 1　ホストの環境

CPU	Core i5-2410M
Memory	4GB
OS	Linux version 3.13.0

表 2　Intel x86 仮想マシンの環境

CPU	Pentium3
Memory	2GB
OS	Linux version 3.13.0

表 3　ARM Cortex-A9 仮想マシンの環境

CPU	Cortex-A9
Memory	2GB
OS	Linux version 3.2.0

表 4　Intel x86 仮想マシンにおける
スナップショットを使った場合の再現性

	最大値（秒）	最小値（秒）	分散
art	7.339520	7.339452	2.34916e-10
equake	1.658267	1.658261	1.06248e-12
hmmer	14.11592	14.11590	6.36993e-12
mpeg2enc	23.58393	23.58392	2.27966e-11

表 5　ARM Cortex-A9 仮想マシンにおける
スナップショットを使った場合の再現性

	最大値（秒）	最小値（秒）	分散
art	2.00465	1.99789	2.98627e-06
equake	0.464201	0.461314	3.84649e-07
hmmer	17.0459	16.9744	4.47751e-04
mpeg2enc	22.6919	22.5681	1.36756e-03

表 6　Intel x86 仮想マシンにおける
スナップショットを使わない場合の再現性

	最大値（秒）	最小値（秒）	分散
art	7.359111	7.339912	7.13139e-06
equake	1.658420	1.658156	2.22141e-09
hmmer	14.12502	14.11685	6.13031e-06
mpeg2enc	23.58551	23.58514	8.92483e-09

表 7　ARM Cortex-A9 仮想マシンにおける
スナップショットを使わない場合の再現性

	最大値（秒）	最小値（秒）	分散
art	2.00287	2.00277	3.18184e-10
equake	0.461295	0.461244	1.0335e-10
hmmer	17.0665	17.0535	1.46352e-05
mpeg2enc	22.5830	22.5655	5.8983e-06

また、スナップショットを用いない場合，Intel x86 仮想マシンでは表 6 より 400 us から 20 ms のばらつきが認められ，ARM Cortex-A9 仮想マシンでは 50 us から 18 ms のばらつきが認められる．Intel x86 仮想マシンではスナップショットを用いた場合にアプリケーション実行時間の再現性が向上するが，ARM Cortex-A9 ではスナップショットを用いない場合に再現性が向上することを確認した．

4.3 QEMU における実行時間変動要因

QEMU による仮想マシンでは計算機システム全体をエミュレートしているため，実機と同様にゲスト環境における割り込みやメモリ，ディスクの状態などの要因によってアプリケーションの実行時間が変動する．これらの要因はスナップショットにより仮想マシンのシステム状態を復元することで再現することが可能であり，その結果としてアプリケーションの実行時間も再現されると考えられる．しかしながら，実際には QEMU 仮想マシンではスナップショットによってもアプリケーションの実行時間は厳密にはできない．この要因として icount の非再現性とゲスト環境の I/O 制御が挙げられ，以下でこれらの要因について述べる．

4.3.1 icount の非再現性

Intel x86 環境でのゲスト OS が参照するハードウェアタイマの値は，QEMU が保持する icount の値によってエミュレートされる．QEMU ではスナップショットによりゲストのシステム状態を復元しても icount の値は復元されず単調増加するため，ハードウェアタイマの値は再現されないことになる．ハードウェアタイマはアプリケーションの実行時間計測だけでなく OS のスケジューリングにも利用されているため，ゲスト OS においてアプリケーション実行後の割り込みのタイミングやプロセスのスケジューリングが変化し，結果として実行時間が再現されないと考えられる．ARM Cortex-A9 の仮想マシンでもハードウェアのタイマは icount によってエミュレートされるため，同様に実行時間が再現されないものと考えられる．このため，スナップショットにより icount の値も復元してハードウェアタイマの時刻も復元することでシステム状態の再現を行い，アプリケーション実行時間の再現性を向上させることができると考えられる．

4.3.2 I/O 制御

QEMU 仮想マシン上で I/O が発行された際，I/O 処理はホストに任されるため，ゲスト上の I/O 処理時間はホストの処理能力に影響される可能性がある．I/O 処理はゲストからホストに遷移し，一方でゲスト上では仮想 CPU が発行済 I/O に代わり別の命令を実行し始め，ゲスト上のシステム時刻も進行する．ホストの I/O 処理が終了してゲストに通知されるが，I/O 処理をホストに任せている間にゲストのシステム時刻が進行しているため，結果としてゲストの I/O 処理時間はホストの I/O 処理時間の影響を受けるこ

とになる．ゲストにおける I/O 処理時間がホストの I/O 処理時間の影響を受ける様子を図 4 に示す．図に示した問題を解決するには，ゲスト OS が I/O を発行して I/O を処理している間，vcpu スレッド を実行しないことで icount とハードウェアタイマの進行を止めるという方法が有効であると考えられる．つまり，worker スレッド と vcpu スレッド を並列ではなく逐次に実行することでこれによりホストの I/O 処理時間による影響を隠ぺいするというものである．ゲスト OS からは I/O の処理時間があたかも 0 秒であるかのように見えることになる．

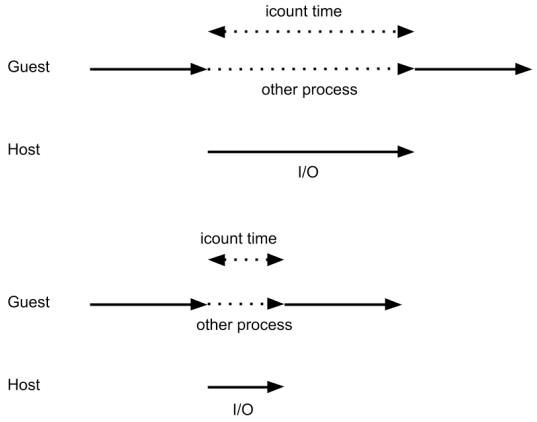

図 4 I/O 処理時間におけるホストからの影響

5. 先行研究

ネットワークシミュレータと計算機システムエミュレータを統合した先行研究として，VMSimInt[3] が知られている．この手法では仮想マシン上におけるアプリケーションの実行時間を 0 秒とし，仮想マシンを 1 台だけ用意してネットワークシミュレータと仮想マシンを逐次的に実行しながら同期を図ることで再現性を確保する．ところが，この方法では仮想マシン上でのアプリケーションの実行時間がシミュレーション時間に組み込まれておらず，通信システムを実験する際に端末の影響を考慮することが難しい．また，仮想マシンを 1 台だけ用意してネットワークシミュレータと逐次的に実行するためシミュレーション全体のパフォーマンスが低下し，多数の端末を前提とした大規模な通信システムへの適用が困難である．

QEMU ではなく Xen とネットワークシミュレータを統合した研究として，SliceTime[4] が知られている．これは仮想マシンとネットワークシミュレータをバリア同期アルゴリズムによって定期的に同期を図ることで，複数台の仮想マシンとネットワークシミュレータの時間進行をそろえるという手法である．本手法では仮想マシンを複数台用意するため，大規模な通信システムを想定した実験が可能となる．しかし，本手法ではバリア同期を利用しているため，同期時刻までは各仮想マシンとネットワークシミュレータがそれぞれ自由に時刻を進めることになり，シミュレーションの再現性が低下するおそれがある．

また，仮想マシン上で動作する OS のカーネルを改変することで時間参照のシステムを変更するという Direct Code Execution[5] という手法が知られている．一般に時間計測では例えば gettimeofday 関数といったシステムコールが利用されるが，本手法ではシステム時刻ではなくシミュレータの時刻を参照するようゲスト環境の OS カーネルを変更する．しかし本手法ではカーネルを書き換える必要があるため，ソースコードの提供されていない OS では評価が不可能であるという問題がある．

6. まとめ

本稿ではネットワークシミュレータと計算機システムエミュレータ QEMU を組み合わせることで，シミュレーションにおけるモデル化の労力と誤差を軽減する手法について議論した．シミュレーションの結果の変化が仮想マシンの外乱ではなくシミュレーションパラメータの変更によるものに限定するため，両者を組み合わせるにあたっては仮想マシンにおけるアプリケーション実行や I/O 処理を再現する必要がある．その再現性を高めるため，スナップショットによりアプリケーションの実行時間を再現する手法を提案した．計算機システムエミュレータ QEMU により Intel x86 の仮想環境と ARM Cortex-A9 の仮想環境を構築し，各々においてアプリケーションを実行してスナップショットによりシステム状態をアプリケーション実行前に復元することでどの程度の再現性が得られるのか評価した．その結果，スナップショットを用いると Intel x86 の環境を持つ仮想マシンにおいて 10 us から 70 us の実行時間のばらつきを確認し，ARM Cortex-A9 の環境を持つ仮想マシンにおいては 3 ms から 124 ms の実行時間のばらつきを確認した．スナップショットを用いても実行時間がばらつく原因として，icount が再現されていないことやゲスト環境における I/O の制御がホストに遷移することが挙げられる．また，スナップショットを用いない場合では Intel x86 の環境を持つ仮想マシンにおいて 400 us から 20 ms の実行時間のばらつきが，ARM Cortex-A9 の環境を持つ仮想マシンにおいては 50 us から 18 ms のばらつきが確認された．

今後の課題として，スナップショットにより icount も復元することで仮想マシンにおけるハードウェアタイマも復元し，ゲスト環境において割り込みの生じるタイミングやプロセスのスケジューリングを再現することが挙げられる．また，worker スレッドと vcpu スレッドを逐次に実行し，worker スレッドの動作中は vcpu スレッドを停止することで，ゲスト環境で発行された I/O 処理がホストで処理される間の時間を隠ぺいする方法も再現性を向上するために有効であると考えられる．Intel x86 のゲスト環境に比べて

ARM Cortex-A9 のゲスト環境のほうが実行時間のばらつきが大きい原因は今後調査し，icount の復元や worker スレッドと vcpu スレッドの逐次実行が ARM Cortex-A9 の仮想マシンにおいても有効であるか確認する．

参考文献

[1] Fabrice Bellard，QEMU，a Fast and Portable Dynamic Translator，2005

[2] QEMU Emulator User Documentation，http://wiki.qemu.org/download/qemu-doc.html，referenced on May 11，2015

[3] Thomas W.，Matthias K.，Mirja K.，Sebastian S.，David W.，VMSimInt: A Network Simulation Tool Supporting Integration of Arbitrary Kernels and Applications，2014

[4] Elias W.，Florian S.，Hendrik v. L.，Tobias H.，Klaus W.，SliceTime: A platform for scalable and accurate network emulation，2011

[5] Daniel C.，Hajime T.，Emilio M.，Mathieu L.，DCE: Test the Real Code of Your Protocols and Applications over Simulated Networks，2014

[6] Standard Performance Evaluation Corporation，http://www.spec.org/，referenced on May 11，2015