Vol.2021-OS-151 No.4
2021/3/1

Arm版XenにおけるVM切替え処理の性能評価

林 海豊[1,a)]　片山 吉章[2,1]　鄭 俊俊[1]　毛利 公一[1]

概要：本論文では，リアルタイムな組込みシステムへ Arm 版 Xen を適用してリアルタイム機能の冗長化を行うべく，Arm 版 Xen における VM 切替え処理の性能評価を行い，その実現可能性が確認できたのでこれについて報告する．組込みシステムでは，ハイパーバイザによる機能統合が行われてきており，同じ機能の VM を複数生成しておくことで，機能の冗長化も可能である．冗長化された機能が機能不全になった際には実行する VM を切り替える必要があり，特にリアルタイム機能を冗長化している場合は，この切替え時間によってデッドラインに間に合わなくなる可能性がある．ハイパーバイザの 1 つである Arm 版 Xen は，リアルタイムシステムや組込みシステム向けの機能拡張や性能改善が行われてきているが，VM の切替えにかかる時間は明らかになっていなかった．

1. はじめに

近年，IoT の普及などによって組込み機器が増加しており，その中には高い信頼性が求められるリアルタイム機能を実行するものも存在している．例えば，自動車では車両制御機能が実行されており，このような機能は障害が発生しても動作継続可能であることが求められる．また，組込みシステムにおいてハイパーバイザが用いられるようになってきている．ハイパーバイザを導入することにより，1 つの計算機上で複数のシステムを動作させることが可能であるため，組込み機器を構成する計算機の数を削減でき，省スペース化や省電力化を実現できる．

以上から，組込みシステムで導入されているハイパーバイザを利用し，リアルタイム機能を冗長化させることが考えられる．冗長化する機能には，あらかじめ稼働系および待機系の仮想計算機（以下，VM）を用意しておく．稼働系の VM で障害が発生した際，待機系の VM で処理を行うように切り替えることで機能の動作を継続させることができる．しかし，この手法は VM 切替え時にオーバヘッドが生じるため，リアルタイム機能を冗長化する場合，VM 切替えがリアルタイム機能のデッドラインに間に合わない可能性がある．

また，現在，Arm 版 Xen[1] と呼ばれるハイパーバイザが組込みシステムに使用され始めている．Arm 版 Xen は，組込みシステムへの適用が考慮されたハイパーバイザであり，リアルタイムシステムや組込みシステム向け機能の開発が行われているが，VM 切替え処理の所要時間は明らかになっていなかった．

以上の背景から，本論文では，Arm 版 Xen によるリアルタイム機能の冗長化を視野に入れ，Arm 版 Xen の VM 切替え処理の性能評価を行った．VM 切替え処理の所要時間を計測し，Arm 版 Xen によってどの程度のデッドラインを持つリアルタイム機能が冗長化可能かを明らかにした．また，VM 切替え処理は Xen やゲスト OS のパラメータに影響される可能性があるため，さまざまなパラメータ設定で計測を行い，VM 切替え処理に対する各種パラメータの影響を確認した．さらに，VM 切替え時に Arm 版 Xen や VM で生じるイベントをトレースし，VM 切替え処理において特に時間がかかっている処理を特定した．

以下，本論文では，2 章で Arm 版 Xen の概要および Xen による機能の冗長化について述べ，3 章で Arm 版 Xen の VM 切替え処理の評価と考察について述べる．また，4 章で今回行った計測の課題と今後の展望について述べ，5 章で本論文の内容をまとめる．

2. Arm版Xen

2.1 概要

Xen は，x86 および Arm アーキテクチャに対応したオープンソースのハイパーバイザである [2]．Xen において，VM はドメインと呼ばれ，特に Xen や他ドメインの管理を行うドメインを dom0，ゲスト OS を動作させて目的の処理を実行するドメインを domU と呼ぶ．dom0 の OS には主

[1] 立命館大学
Ritsumeikan University
[2] 三菱電機株式会社情報技術総合研究所
Information Technology R&D Center, Mitsubishi Electric Corporation
[a)] krin@asl.cs.ritsumei.ac.jp

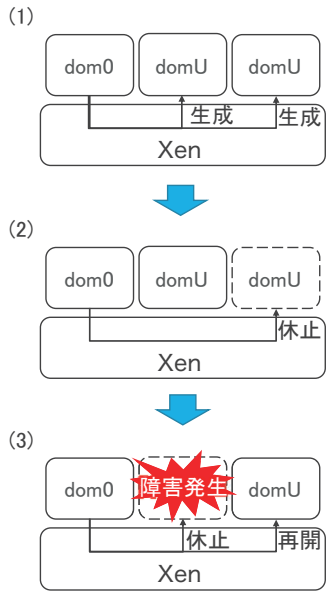

図 1 Xen における機能の冗長化手法

に Linux が用いられ，domU の OS には Linux や Windows などが用いられる．Xen で利用可能な仮想化方式は，完全仮想化と準仮想化の他に，完全仮想化と準仮想化を混合させた方式も用意されている．混合方式では，完全仮想化のように CPU の仮想化支援機能を使用し，準仮想化のようにデバイスエミュレーションを行わないことで性能の向上を図っている．

Arm 版 Xen は，特に Arm アーキテクチャで実行する Xen を指す．Arm 版 Xen は，x86 版 Xen と同等の機能を有しているが，利用可能な仮想化方式は PVH と呼ばれる混合方式のみである．また，Arm 版 Xen のコードサイズは，x86 版 Xen の約 6 分の 1 である．コードサイズが小さいことにより，Arm 版 Xen を高信頼システムに導入する際の安全性検証コストを抑えることが可能となっている．

また，近年，Xen ではリアルタイムシステムや組込みシステム向けの機能が多数開発されている [3], [4]．例えば，null スケジューラと呼ばれるドメインスケジューラが挙げられる．ドメインスケジューラは，Xen において各ドメインが持つ仮想 CPU（以下，VCPU）のスケジューリング方式を表現する．null スケジューラは，VCPU と物理 CPU（PCPU）を 1 対 1 で対応させる静的なスケジューラであり，スケジューリングのオーバヘッドを最小化できるため，リアルタイムシステムでの活用が期待されている．また，Xen では dom0 のデバイスドライバを利用してデバイスを操作するため，Xen はデバイスドライバを持たない．したがって，Xen を他の計算機に移植する場合，移植先の計算機で実行可能な OS を dom0 とすることで移植先デバイスに対応できるため，Arm 版 Xen と組込みシステムとの親和性は高い．

2.2 Xen による機能の冗長化

Xen による機能の冗長化手法を**図 1** に示し，手順を以下に示す．

(1) Xen および dom0 を起動した後，dom0 は，Xen を通して冗長化する機能に対して稼働系と待機系の domU を生成する．
(2) 稼働系の domU は，初期化した後，対象機能の実行を始める．また，待機系の domU は，初期化した後，dom0 から Xen を通して休止させる．
(3) 稼働系に障害が発生した際，dom0 は Xen を通して稼働系を休止させ，待機系を再開させて機能の実行を再開させる．

このように，Xen において動作させる VM を切り替える際には，dom0 から domU の休止および再開を行う．domU の休止処理は，Xen が対象の domU を CPU スケジュール対象から除外することで実現しており，再開処理では Xen が対象の domU を再び CPU スケジューリング対象に含めることで実現している．そのため，もし稼働系の domU が制御不能となった場合でも，確実に休止させることができる．また，待機系の domU を初期化した状態で休止させておくことで，VM 切替え後すぐに機能を実行できる．しかし，Arm 版 Xen において VM の切替えにかかる時間が明らかになっておらず，どの程度のデッドラインを持つリアルタイム機能が冗長化できるか不明であった．

3. VM 切替え処理の性能評価

3.1 実験環境

Arm 版 Xen における VM 切替え処理の評価実験では，Raspberry Pi 4 Model B と PC の 2 つの計算機を使用した．Raspberry Pi 4 Model B は Arm アーキテクチャのシングルボードコンピュータであり，Arm 版 Xen を実行するために使用した．PC は，Raspberry Pi 4 Model B で実行している Arm 版 Xen を操作するために使用した．PC のハードウェアとソフトウェアの構成を**表 1** に示す．

また，本実験では，Raspberry Pi 4 Model B で実行する Arm 版 Xen および Linux のバージョンが異なる 2 つの実験環境において性能評価を行った．以下，2 つの実験環境を，それぞれ旧環境，新環境と呼ぶ．旧環境および新環境のハードウェアとソフトウェアの構成をそれぞれ**表 2**, **表 3** に示す．

旧環境は，Arm 版 Xen 4.13.0，dom0 カーネルに Linux 4.19.118 を用いた実験環境である．旧環境では，Linux の Xen ドライバや Arm 版 Xen が，Raspberry Pi 4 Model B のデバイスや DMA アドレスの仕様に一部対応していなかった．例えば，Raspbeery Pi 4 Model B にはメモリ量が 1GB よりも多いモデルがあるが，Raspbeery Pi 4 Model B の一部のデバイスは下位 1GB のメモリにのみアクセス可能である．これにより，旧環境で Arm 版 Xen は実行可

表 1　PC の構成

CPU	Intel Core i5 9400F
メモリ	24GB
ストレージ	1TB SSD
OS	Ubuntu 18.04.5 LTS + Linux 4.15.0

表 2　旧環境の構成

CPU		Arm Cortex-A72（4 コア）
メモリ		4GB
ストレージ		64GB SD カード
ハイパーバイザ		Arm Xen 4.13.0
OS	dom0	Ubuntu 18.04.4 LTS + Linux 4.19.118
	domU	Ubuntu 18.04.4 LTS + Linux 4.14.18

表 3　新環境の構成

CPU		Arm Cortex-A72（4 コア）
メモリ		4GB
ストレージ		32GB SD カード
ハイパーバイザ		Arm Xen 4.14.1
OS	dom0	Debian GNU/Linux 10 + Linux 5.9.0
	domU	Ubuntu 18.04.5 LTS + Linux 4.15.18

能であるものの，USB ポートが使用できない，domU に割当て可能なメモリが少ないといった問題が生じていた．この問題は，2020 年 9 月 29 日に，Arm 版 Xen と Linux のアップデートによって解決されたことが報告された [5]．

新環境は，上記の問題が解決された Arm 版 Xen 4.14.1 と Linux 5.9.0 を用いた実験環境である．新環境においても VM 切替え処理の評価実験を行うことで，Arm 版 Xen および Linux のアップデートの影響を確認した．

3.2　VM 切替え処理の計測方法

Xen において，VM の切替えを行うためには，dom0 から Xen にハイパーコールを行い，domU の休止と再開を行う必要がある．dom0 から domU の休止や再開のハイパーコールを発行する方法としては，以下のものが挙げられる．

カーネルモジュールの作成

dom0 のカーネルモジュールを作成し，そこから独自に domU の休止や再開のハイパーコールを行う方法．

libxenctrl の利用

Xen では libxenctrl と呼ばれるユーザライブラリが用意されている [6]．libxenctrl には，ドメインに存在する Xen ドライバにハイパーコールを依頼する関数が定義されているため，それらの関数を用いてユーザプログラムから domU の休止や再開のハイパーコールを行う方法．

xl コマンドの実行

dom0 から Xen の管理や操作を行うための xl コマンドを用いる方法 [7]．xl コマンドには domU の休止および再開を行うためのサブコマンドが用意されており，これを実行することで domU の休止や再開のハイパーコールを行うことができる．

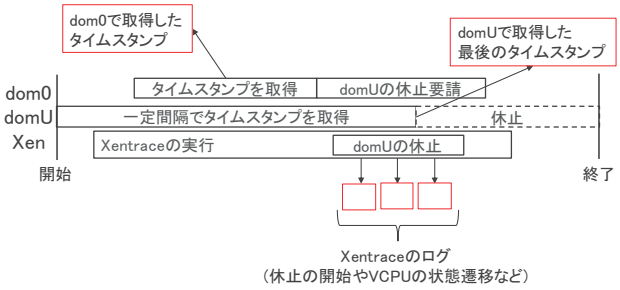

図 2　domU の休止処理の計測手順

本実験では，Arm 版 Xen による VM 切替えの性能を確認するため，xl コマンドを利用した際の Arm 版 Xen における domU の休止・再開処理を計測した．xl コマンドによる domU の休止・再開処理の概要を以下に示す．なお，基本的には休止処理についての説明であり，再開処理で行う処理は括弧内に示す．

(1) dom0 において，xl pause（xl unpause）コマンドを実行し，domU の休止（再開）を開始する．
(2) Xen において，対象の domU をスケジューリングから除外（スケジューリングに追加）する．
(3) domU が休止（再開）される．

このように，domU の休止・再開処理は，dom0，domU，Xen にまたがって行われる．そこで，domU の休止・再開処理の総所用時間だけでなく，dom0，domU，Xen において処理をトレースし，domU の休止・再開処理の所用時間を処理ごとに分割して計測した．

Xen 内部の計測については，Xentrace と呼ばれるトレーサが用意されている [8]．Xentrace は，Xen で生じたイベントおよびイベントが生じたときのタイムスタンプが収集可能であるため，これを用いた．Xentrace のタイムスタンプには，Arm の Generic Timer と呼ばれるハードウェアカウンタの値が用いられている．この値は，CNTPCT_EL0 と呼ばれるレジスタにアクセスすることで取得可能である [9]．そのため，dom0 や domU でトレースする際にも CNTPCT_EL0 の値をタイムスタンプとすることで，各トレースログ間の所要カウントが計算可能である．また，CNTFRQ_EL0 と呼ばれるレジスタから，Generic Timer がカウントアップする周波数を取得することが可能であるため，各トレースログ間の所要カウントを CNTFRQ_EL0 の値で割ることで各トレースログ間の所要時間を計算することができる．

domU の休止処理の性能計測手順をタイムラインで表したものを**図 2** に示し，計測手順の詳細を以下で述べる．なお，再開処理も休止処理と同様の方法で計測する．

(1) 休止させる domU において，一定間隔でタイムスタンプを取得する．本実験では，取得間隔を 4ms とした．
(2) Xentrace を実行し，Arm 版 Xen で生じるイベントを

表 4 パラメータとその設定パターン

	パラメータ	基準設定	変更先設定
domU	VCPU 数	3	1
	メモリ量	813MB	155MB
	ストレージの種類	論理パーティション	ファイル
	CPU 負荷	なし	4 プロセス分
	ネットワーク負荷	なし	1000Mbps
dom0	CPU 負荷	なし	4 プロセス分
	ネットワーク負荷	なし	1000Mbps
	xl コマンドの nice 値	0（通常優先度）	-20（高優先度）
	xl コマンドのスケジューリングポリシ	SCHED_OTHER（通常優先度）	SCHED_FIFO（高優先度）
Xen	domU の数	1 つ	2 つ
	ドメインスケジューラ	credit2 スケジューラ	null スケジューラ

表 5 domU の再開・休止処理の統計結果 (µs)

変更したパラメータ		休止処理の総所要時間				再開処理の総所要時間			
		平均値	分散	最大値	最小値	平均値	分散	最大値	最小値
	基準設定	14,777	328,259	15,726	14,170	16,579	1,048,055	18,386	15,641
domU	VCPU 数	20,479	626,909	21,547	19,549	22,216	4,447,115	26,394	20,741
	メモリ量	15,063	1,553,065	17,409	14,021	15,010	258,884	15,645	14,216
	ストレージの種類	21,696	1,009,063	22,486	19,976	20,756	1,300,465	22,046	18,987
	CPU 負荷	13,491	11,830	13,677	13,377	21,941	113,732,013	43,261	16,171
	ネットワーク負荷	44,670	45,852,362	54,794	36,443	14,144	44,723	14,462	13,942
dom0	CPU 負荷	65,701	161,189,947	87,675	57,629	61,796	13,880,921	66,159	57,685
	ネットワーク負荷	63,176	1,559,388,900	141,991	41,376	37,528	3,465,128	40,112	34,467
	xl コマンドの nice 値	17,106	1,357,347	18,582	15,744	16,999	605,749	18,115	15,716
	xl コマンドのスケジューリングポリシ	26,481	401,246,355	66,512	15,407	16,514	79,679	16,887	16,069
Xen	domU の数	14,475	229,106	15,015	13,801	14,862	852,536	16,536	14,137
	ドメインスケジューラ	25,402	111,154,445	46,368	18,848	19,727	329,047	20,615	19,050

トレースする．

(3) dom0 でタイムスタンプを取得し，xl コマンドを用いて Arm 版 Xen に休止を要請する．このとき取得したタイムスタンプを休止処理の開始時刻とする．

(4) domU が休止し，タイムスタンプの取得が中断される．このとき最後に取得されたタイムスタンプを，休止処理の完了時刻とする．

(5) 取得したタイムスタンプを用いて，各処理間の所要時間を計算する．

3.3 旧環境での domU の休止・再開処理の計測

3.3.1 実験方法

domU の休止・再開処理がどのようなパラメータの影響を受けるのかを検証するため，旧環境において Arm 版 Xen やドメインのパラメータを変更しながら，xl コマンドによる domU の休止・再開処理の所要時間を計測した．また，domU の休止・再開処理の中でどの処理に時間がかかっているかを確認するため，domU の休止・再開を主要な処理ごとに細分化して計測を行った．

パラメータとその設定パターンを**表 4** に示す．パラメータの基準設定，および基準設定から変更する際の設定は，例えば，domU の VCPU 数の場合，基準設定は 3 つとし，基準設定から変更する場合は 1 つとした．なお，CPU 負荷の生成には stress コマンドを利用した．また，ネットワーク負荷は，PC から Raspbeerry Pi 4 Model B に対して iperf[10] を用いて生成した．3.2 節に示した計測方法で，表 4 の基準設定と，基準設定からいずれか 1 つのパラメータを変更したパターンを 5 回ずつ計測した．

3.3.2 実験結果と考察

各パラメータパターンにおける domU の休止・再開処理の総所要時間の平均値，分散，最大値，最小値を**表 5** にまとめた．dom0 に CPU 負荷やネットワーク負荷をかけた場合，domU の休止・再開処理の平均値，分散，最大値，最小値が大きい傾向にある．休止処理の最大値は，dom0 に 1000Mbps のネットワーク負荷をかけた場合の 142ms であり，再開処理の最大値は，dom0 に 4 プロセス分の CPU 負荷をかけた場合の 66ms である．これらの時間よりも短いデッドラインをもつリアルタイム機能は，Arm 版 Xen の VM 切替えが間に合わないため，冗長化できない．この原因は未だ明らかになっていないが，dom0 に負荷をかけることで，xl コマンドのプロセス生成やスケジューリングなどの処理が遅延されているのではないかと考えている．実際，dom0 に負荷をかけた状態で，nice 値やスケジューリングポリシを用いて xl コマンドの優先度を上げたところ，domU の休止・再開の所要時間が 18ms から 55ms までに抑えられることが確認できた．今後，dom0 カーネルのス

表 6 休止処理の平均所要時間 (µs)

変更したパラメータ		xl コマンド実行〜 domU の休止	domU の休止〜 Xen で休止開始	Xen で休止開始〜 VCPU の休止	総所用時間
	基準設定	10,709	2,548	1,520	14,777
domU	VCPU 数	16,411	2,070	1,998	20,479
	メモリ量	10,995	2,559	1,509	15,063
	ストレージの種類	17,629	1,149	2,918	21,696
	CPU 負荷	11,806	1,657	28	13,491
	ネットワーク負荷	39,414	1,955	3,301	44,670
dom0	CPU 負荷	56,518	1,416	7,766	65,701
	ネットワーク負荷	48,137	1,934	13,105	63,176
	xl コマンドの nice 値	13,038	2,307	1,761	17,106
	xl コマンドのスケジューリングポリシ	22,413	2,556	1,512	26,481
Xen	domU の数	10,407	3,207	861	14,475
	ドメインスケジューラ	21,334	2,547	1,520	25,402

表 7 再開処理の平均所要時間 (µs)

変更したパラメータ		xl コマンド実行〜 Xen で再開開始	Xen で再開開始〜 VCPU の再開	VCPU の再開〜 domU の再開	総所用時間
	基準設定	13,821	17	2,741	16,579
domU	VCPU 数	19,022	31	3,162	22,216
	メモリ量	13,898	41	1,072	15,010
	ストレージの種類	19,013	40	1,703	20,756
	CPU 負荷	19,132	40	2,769	21,941
	ネットワーク負荷	13,831	16	297	14,144
dom0	CPU 負荷	60,601	39	1,156	61,796
	ネットワーク負荷	36,689	16	823	37,528
	xl コマンドの nice 値	15,875	18	1,107	16,999
	xl コマンドのスケジューリングポリシ	15,599	60	854	16,514
Xen	domU の数	14,063	66	732	14,862
	ドメインスケジューラ	18,908	19	799	19,727

ケジューリングなどを改良することで，よりデッドラインが短いリアルタイム機能を Arm 版 Xen で冗長化できる可能性がある．

また，domU にネットワーク負荷をかけた場合，domU の休止処理の所要時間が大きい傾向にある．Arm 版 Xen の domU は，すべて PVH と呼ばれる仮想化方式で実現される．PVH の domU は dom0 にデバイスの処理を依頼するため，Arm 版 Xen で domU にネットワーク負荷をかけた場合，同時に dom0 にも負荷がかかる．そのため，domU にネットワーク負荷をかけた場合も，dom0 において xl コマンドのプロセス生成などが遅延されているのではないかと推察される．一方，domU の再開処理の所要時間は基準設定と同等である．domU を再開する際，ネットワーク負荷がかけられている domU は休止されている．そのため，dom0 や Xen が domU のネットワーク処理を省略し，負荷を低減している可能性がある．

また，トレースログを用いて細分化した domU の休止・再開の平均所要時間を，**表 6** および**表 7** に示す．表は，各パラメータにおける，区間ごとの所要時間と総所要時間を表す．dom0，domU，Xen のトレースログから，domU の休止・再開処理は 3 つの区間に分けることができた．休止処理は，まず dom0 で xl コマンドの実行が行われ，domU が休止した．その後，Xen で domU の休止処理が開始され，domU の VCPU が休止状態に遷移した．また，再開処理は，まず dom0 で xl コマンドの実行が行われ，Xen で domU の再開処理が開始された．その後，domU の VCPU が実行状態に遷移し，domU が再開する．

休止処理では，どのパラメータパターンにおいても，dom0 で xl コマンドを実行してから domU が休止するまでの区間が最も長かった．また，再開処理では，どのパラメータパターンにおいても，dom0 で xl コマンドを実行してから Xen で domU の再開処理が開始するまでの区間が最も長かった．両区間は，総所要時間の 70%から 98%を占めている．また，総所用時間が長いほど，両区間の所用時間も長い傾向にあるため，両区間は所要時間延長の主な原因となっていることが分かる．したがって，domU の休止処理や再開処理の所用時間を改善する場合，dom0 で xl コマンドを実行してから次の処理までの区間を改善することが最も効果的である．

3.4 新環境と旧環境での domU の休止・再開処理の比較

3.4.1 実験方法

新環境において domU の休止・再開処理の所要時間を計測し，Arm 版 Xen と Linux のアップデートの影響を確認

表 8 新環境と旧環境の比較において計測したパラメータパターン

	パラメータ	基準設定	変更先設定
domU	VCPU 数	1	
	メモリ量	813MB	
	ストレージの種類	ファイル	
	CPU 負荷	なし	
	ネットワーク負荷	なし	
dom0	CPU 負荷	なし	4 プロセス分
	ネットワーク負荷	なし	1000Mbps
	xl コマンドの nice 値	0(通常優先度)	
	xl コマンドのスケジューリングポリシ	SCHED_OTHER(通常優先度)	
Xen	domU の数	1 つ	
	ドメインスケジューラ	null スケジューラ	

した．なお，新環境の結果と比較するため，旧環境においても同じパラメータパターンで domU の休止・再開処理の所要時間を計測した．新環境と旧環境において domU の休止・再開処理を計測したパラメータパターンを**表 8** に示す．本実験で計測したパラメータパターンは，基準設定，および dom0 に CPU 負荷やネットワーク負荷をかけた場合である．3.3.2 項の実験結果では，旧環境で dom0 へ CPU 負荷やネットワーク負荷をかけた場合，domU の休止処理や再開処理の所要時間が最大値をとっていた．新環境においてもこれらのパラメータパターンで計測を行い，所要時間の最大値が改善されているかを確認した．3.2 節に示した計測方法で，表 8 のパラメータパターンに対して xl コマンドによる domU の休止・再開処理の所要時間を 50 回ずつ計測した．

3.4.2 実験結果と考察

計測結果を散布図としてプロットしたものを**図 3** に示す．横軸が domU の休止処理の所用時間を表し，縦軸が domU の再開処理の所用時間を表している．各点は実験環境とパラメータパターンによって色分けされており，1 点が 1 回の休止および再開の計測結果を表している．

新環境において dom0 に CPU 負荷やネットワーク負荷をかけた場合，旧環境と同様に休止・再開処理の所要時間が長くなる傾向が見受けられる．ただし，新環境の方が分散が小さく，旧環境に比べて所要時間が短い傾向にあることが分かった．一方，旧環境において dom0 に負荷をかけた際，再開処理に 2s 以上かかった場合が 3 回計測された．この原因は未だ解明できていないが，旧環境でリアルタイム機能を冗長化することは困難であることが分かった．

また，新環境における domU の休止・再開処理の平均所要時間をトレースログで細分化したものを**表 9** および**表 10** に示す．表 9 より，新環境の休止処理は，旧環境と同様に dom0 が xl コマンドを実行してから domU が休止するまでの区間が最も長く，総所用時間の 86%から 97%を占めていた．また，休止処理の平均所要時間は旧環境に比べて短くなっており，特に dom0 にネットワーク負荷をかけた場合の休止処理の所要時間は，旧環境と比べて 28ms

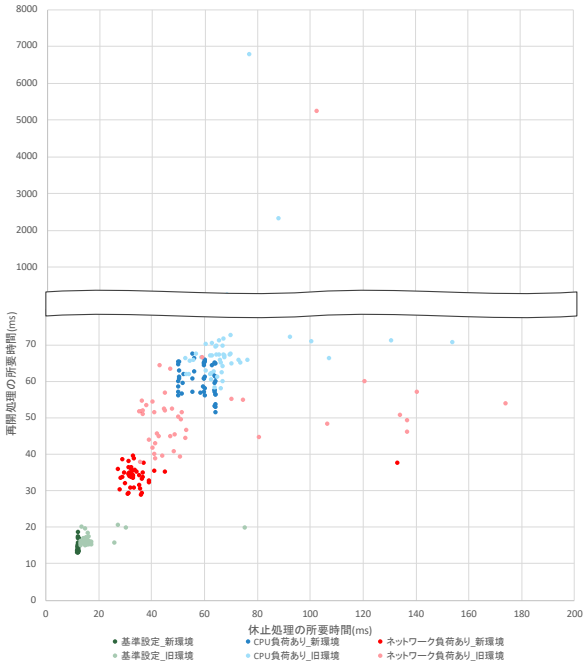

図 3 新環境および旧環境における休止・再開の所要時間

程度短縮されている．また表 10 より，新環境の再開処理では，dom0 が xl コマンドを実行してから Arm 版 Xen で再開処理が開始されるまでの区間が最も長く，総所用時間の 87%から 97%を占めていた．以上の結果から，新環境は旧環境と比べて，domU の休止・再開処理の所要時間の平均と分散が改善しており，より多くのリアルタイム機能に対して VM 切替え可能であることが分かった．また，新環境においても，dom0 が xl コマンドを実行してから次の処理までの区間が，domU の休止や再開処理の所要時間の多くを占めていることが分かった．

3.5 ハイパーコールの評価

これまでの計測結果から，domU の休止・再開処理の中で最も所要時間が長い区間は，dom0 が xl コマンドを実行してから次の処理が行われるまでの区間である．この区間は，xl コマンドの実行などといった dom0 における処理と，dom0 から Xen へのハイパーコール処理で構成されて

表 9 新環境における休止処理の平均所要時間 (µs)

変更したパラメータ		xl コマンド実行〜 domU の休止	domU の休止〜 Xen で休止開始	Xen で休止開始〜 VCPU の休止	総所用時間
基準設定		10,490	1,628	23	12,141
dom0	CPU 負荷	56,578	1,862	22	58,461
	ネットワーク負荷	33,559	1,690	22	35,272

表 10 新環境における再開処理の平均所要時間 (µs)

変更したパラメータ		xl コマンド実行〜 Xen で再開開始	Xen で再開開始〜 VCPU の再開	VCPU の再開〜 domU の再開	総所用時間
基準設定		12,473	16	1,873	14,362
dom0	CPU 負荷	58,886	16	1,727	60,630
	ネットワーク負荷	32,853	15	1,699	34,568

いる．既に区間ごとの所要時間が分かっているため，dom0 における処理とハイパーコールの所要時間のどちらかを計測することで，他方の所要時間も間接的に把握することが可能である．今回は，より計測が簡単なハイパーコールを計測し，dom0 における処理とハイパーコールの所要時間を確認した．

3.5.1 実験方法

これまでの実験では xl コマンドによる domU の休止・再開処理を計測していた．xl コマンドでは，libxenctrl と呼ばれるライブラリを用いてハイパーコールが行われる．本実験では xl コマンドのハイパーコール処理の所用時間を確認したいため，新環境において libxenctrl によるハイパーコールの所要時間を計測した．

ハイパーコール処理の中でも，特に dom0 が Xen に処理を依頼する時間を確認したいため，Xen で行う処理ができるだけ少ないハイパーコールで計測を行う必要がある．本実験では，version ハイパーコールと呼ばれるハイパーコールを用いて計測を行った．version ハイパーコールは，Xen のメジャーバージョンおよびマイナーバージョンを返す単純なハイパーコールであるため，今回の計測に適している．

これまでの実験結果から，dom0 に CPU 負荷やネットワーク負荷をかけると休止・再開処理の所要時間が最大値を取るということが分かっている．そこで，表 8 のパラメータパターンに対して，新環境で version ハイパーコールを 1000 回ずつ計測した．version ハイパーコールの計測手順を以下に示す．なお，ハイパーコールの開始時刻と終了時刻には CNTPCT_EL0 の値を使用した．

(1) version ハイパーコールの開始時刻を取得する．
(2) libxenctrl の関数を用いて version ハイパーコールを呼ぶ．
(3) version ハイパーコールの終了時刻を取得する．
(4) version ハイパーコールの所要時間を計算し，表示する．

3.5.2 実験結果と考察

計測結果をパラメータパターンごとにプロットしたものを**図 4** に示す．横軸は，version ハイパーコールの所用時間の対数軸であり，縦軸は各パラメータパターンを表している．また，パラメータパターンによって点が色分けされており，1 点が 1 回の計測結果を表している．

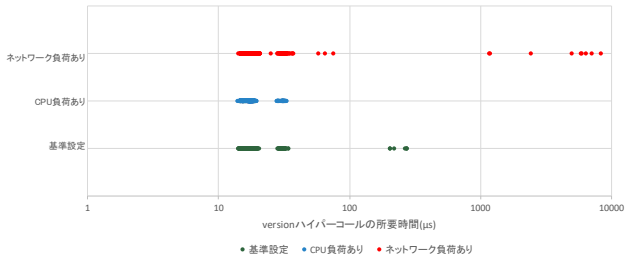

図 4 version ハイパーコールの所要時間

いずれのパラメータパターンにおいても，version ハイパーコールの所要時間は 10µs オーダーに集中しており，全体の 98%以上を占めていた．また，基準設定の場合，0.7%の割合で 100µs オーダーの所要時間がかかる場合があった．一方，dom0 に CPU 負荷をかけた場合，version ハイパーコールの遅延は見受けられず，むしろ所要時間が 100µs 以上かかる場合がなくなった．この原因はよく分かっていないが，CPU 負荷によって dom0 のスケジューリングが変化したことが影響していると推察される．また，dom0 にネットワーク負荷をかけた場合，100µs オーダーの所要時間がかかる場合はなく，0.9%の割合で 1000µs オーダーの所要時間がかかる場合が見受けられた．このことから，version ハイパーコールで 100µs 以上かかる場合，dom0 のデバイス周りの負荷が影響していると考えられる．

これまでの計測結果より，dom0 の xl コマンド実行から次の処理までの所要時間は 10ms のオーダーであった．version ハイパーコールの所要時間は 10µs から 1ms のオーダーであるが，実際には 10µs オーダーの場合がほとんどであった．したがって，dom0 の xl コマンド実行から次の処理までの区間において，ハイパーコール処理が占める割合は小さく，dom0 における処理がほとんどを占めていることになる．また，ハイパーコールの所要時間は，外れ値の場合を除き，dom0 への負荷によって延長されない．そ

のため，domU の休止・再開の所要時間が延長されている主な原因も，dom0 における処理であると推察される．

4. 課題と今後の展望

旧環境において各パラメータの影響を確認した実験では，各パラメータパターンを 5 回ずつしか計測していない．分散が大きいパラメータパターンも見受けられたため，さらに複数回計測を行うことでより正確なパラメータの影響を確認できる可能性がある．また，本実験で計測したパラメータパターンは，基準設定と基準設定から 1 つのパラメータを変更した場合であるため，網羅的ではない．本実験で変更の影響が小さかったパラメータであっても，それらを複数同時に変更した場合，domU の休止・再開処理の所要時間が大きく変化する可能性がある．

xl コマンドによる domU の休止・再開処理の計測において，domU のタイムスタンプ取得間隔を 4ms とした．そのため，domU のタイムスタンプには最大 4ms の誤差が存在する．タイムスタンプ取得間隔を短くすることで誤差は小さくできるが，かわりに計測のオーバヘッドは増大する．

本論文では，xl コマンドや libxenctrl を用いた domU の休止・再開処理を計測したが，自作カーネルモジュールからハイパーコールを行う場合は計測していない．カーネルモジュールから直接 domU の休止・再開処理を行う場合，ユーザランドの処理を経由しないため，所要時間が短い可能性が高い．そのため，今後カーネルモジュールから直接 domU の休止・再開処理を行う場合を計測し，所要時間を確認する必要がある．

実験結果から，Arm 版 Xen における VM 切替え処理の内，dom0 における xl コマンドの実行から次の処理までの区間に時間がかかることが分かった．特に，dom0 上の処理に多くの時間がかかっていたため，xl コマンドの生成やスケジューリングに時間が費やされている可能性がある．そのため，VM 切替え時のプロセス生成やスケジューリングを改善することで，VM 切替えの所要時間が改善できる可能性がある．例えば，VM 切替えを行うプログラムを常駐させることでプロセス生成を回避したり，dom0 のスケジューリング処理を改善することが考えられる．今後は，dom0 上の処理の内，どの処理に時間がかかっているかを確認した後，VM 切替え処理の改善を行っていく．

5. おわりに

本論文では，Arm 版 Xen においてリアルタイム機能の冗長化を行うため，Arm 版 Xen の VM 切替え処理について性能評価を行った．実験結果から，dom0 に CPU 負荷やネットワーク負荷がかかるような状況においては domU の休止や再開の所要時間が長くなるため，VM 切替え処理に時間がかかることが分かった．旧環境では，dom0 の負荷によって再開処理に 2s 以上かかる場合があるため，旧環境においてリアルタイム機能を冗長化することは困難である．一方，Arm 版 Xen 4.14.1 および Linux 5.9.0 を用いた新環境では，休止・再開処理の平均や分散が小さくなっており，旧環境と比較して，よりデッドラインが短いリアルタイム機能を冗長化可能であることが分かった．

また，Arm 版 Xen における VM 切替え処理の内，dom0 における xl コマンドの実行から次の処理までの時間が長いことが分かった．特に，dom0 上の処理に多くの時間がかかっており，dom0 に負荷をかけることでさらに長くなる傾向にあることが分かった．

参考文献

[1] Xen Project: Xen ARM with Virtualization Extensions whitepaper, Xen Project Wiki (online), available from ⟨https://wiki.xenproject.org/wiki/Xen_ARM_with_Virtualization_Extensions_whitepaper/⟩ (accessed 2021-01-26).
[2] Barham, P., Dragovic, B., Fraser, K. et al.: Xen and the Art of Virtualization, *SIGOPS Oper. Syst. Rev.*, Vol. 37, No. 5, pp. 164–177 (2003).
[3] Xen Project: Xen Project 4.9 Feature List, Xen Project Wiki (online), available from ⟨https://wiki.xenproject.org/wiki/Xen_Project_4.9_Feature_List/⟩ (accessed 2021-01-26).
[4] Xen Project: Xen Project 4.12 Feature List, Xen Project Wiki (online), available from ⟨https://wiki.xenproject.org/wiki/Xen_Project_4.12_Feature_List/⟩ (accessed 2021-01-26).
[5] Stabellini, S. and Shaposhnik, R.: Xen on Raspberry Pi 4 adventures, Xen Project (online), available from ⟨https://xenproject.org/2020/09/29/xen-on-raspberry-pi-4-adventures/⟩ (accessed 2021-01-26).
[6] Campbell, I.: Xen toolstack library API/ABIs, Xen Project (online), available from ⟨https://xenbits.xen.org/people/liuw/libxenctrl-split/vwip.html⟩ (accessed 2021-02-03).
[7] Xen Project: XL, Xen Project Wiki (online), available from ⟨https://wiki.xenproject.org/wiki/XL⟩ (accessed 2021-01-28).
[8] Faggioli, D.: Tracing with Xentrace and Xenalyze, Xen Project (online), available from ⟨https://xenproject.org/2012/09/27/tracing-with-xentrace-and-xenalyze/⟩ (accessed 2021-01-28).
[9] Arm Ltd.: *ARM Cortex-A72 MPCore Processor Technical Reference Manual*, Arm Developer (2016).
[10] Distributed Applications Support Team (DAST) of the National Laboratory for Applied Network Research (NLANR): iPerf - The ultimate speed test tool for TCP, UDP and SCTP, iPerf (online), available from ⟨https://iperf.fr/⟩ (accessed 2021-01-27).

本論文に掲載している製品名等は，各社の商標または登録商標です．