Vol.2023-OS-158 No.28
2023/2/22

HPCクラウドにおける割り込み処理による OSノイズの影響の評価

西本 伊織[1]　小林　諭[2]　山内 利宏[2]　加藤 純[3]　佐藤 充[3]　谷口 秀夫[2]

概要：近年，大規模計算の需要が高まり，HPC（High Performance Computing）とクラウドが融合したHPC クラウドが注目されている．HPC クラウドでは，HPC 用のハードウェアアクセラレータやネットワークファブリックを前提としたHPC クラウド専用のインフラを構築・運用している．このようなHPC クラウド専用のインフラは，汎用的なクラウドのインフラに比べ高コストである．これに対し，汎用的なクラウドのインフラでHPC アプリケーションの性能が十分発揮されるのであれば，HPC クラウドをより低コストで運用できる．しかし，汎用的なクラウドのインフラでHPC アプリケーションを実行する場合，計算処理の性能を十分発揮する上では，OS ノイズが課題となる．OS ノイズは，HPC に特有の課題であり，並列処理の同期の遅延によりアプリケーションの性能低下を引き起こす．OS ノイズの原因の 1 つとして，通信による割り込み処理がある．そこで，本稿では，割り込み処理によるOS ノイズがアプリケーションに与える影響を評価する．空きコアがある環境における評価では，割り込みを空きコアに集中させることにより，アプリケーションの性能が既存の割り込み制御手法を利用した場合に比べ，最大5%改善することを示した．また，既存の割り込み手法はいずれも割り込みの制御が有効に行われていないことを示し，空きコアがない環境でOS ノイズを削減するような新しい割り込み制御手法の必要性と課題を示した．

IORI NISHIMOTO[1]　SATORU KOBAYASHI[2]　TOSHIHIRO YAMAUCHI[2]　JUN KATO[3]
MITSURU SATO[3]　HIDEO TANIGUCHI[2]

1. はじめに

近年，創薬向け分子動力学計算，材料向け量子科学計算，およびAI 処理など大規模計算の需要が高まり，HPC（High Performance Computing）とクラウドが融合したHPC クラウド [1], [2], [3] が注目されている．HPC クラウドでは，HPC 用のハードウェアアクセラレータやネットワークファブリックを前提としたHPC クラウド専用のインフラを構築・運用している．このようなHPC を前提としたHPC クラウド専用のインフラは，汎用的なクラウドのインフラに比べ高コストである．これに対し，汎用的なクラウドのインフラでHPC アプリケーションの性能が十分発揮されるのであれば，HPC クラウドをより低コストで運用できる．

しかし，汎用的なクラウドのインフラでHPC アプリケーションを実行する場合，計算処理の性能を十分発揮する上では，OS ノイズが課題となる．OS ノイズは，HPC に特有の課題であり，並列処理の同期の遅延によりアプリケーションの性能低下を引き起こす．このため，HPC クラウドの計算処理におけるOS ノイズを明らかにし，OS ノイズによるHPC アプリケーションへの影響を明らかにする必要がある．また，OS ノイズを削減するOS レベルの制御法が必要である．

OS ノイズの原因の 1 つとして，割り込み処理がある．特定のCPU コア上でアプリケーションの実行中に割り込みが発生すると，アプリケーションの処理を一時中断し，割り込み処理を行う．割り込み処理が特定のコアに集中すると，CPU 処理時間を割り込み処理に奪われるため，当該コアでのアプリケーションの実行に影響が出てしまう．特に，HPC アプリケーションの場合，並列処理における同期の遅延が生じる要因になるため，OS ノイズの削減は重要な課題である．

そこで，本研究の目的は，汎用的なクラウドのインフラ

[1] 岡山大学大学院自然科学研究科
Graduate School of Natural Science and Technology, Okayama University
[2] 岡山大学学術研究院自然科学学域
Faculty of Natural Science and Technology, Okayama University
[3] 富士通株式会社
Fujitsu Limited

で HPC クラウドを実現した際に，割り込み処理による OS ノイズを削減する制御法を実現することである．このために，まず，汎用的なクラウドのインフラで HPC アプリケーションを実行した場合を想定して，Linux の既存の割り込み制御法により割り込み先コアを制御し，この結果，割り込み処理による OS ノイズが HPC アプリケーションに与える影響を明らかにする．次に，HPC アプリケーションが使用しない CPU コア（以降，空きコア）がある場合について，静的に割り込み処理を空きコアで処理した場合の結果を評価し，割り込み処理による OS ノイズをうまく制御できた場合と，既存の制御手法を比較評価し，割り込み制御手法の課題を示す．最後に，HPC アプリケーションを実行する HPC クラウド環境向けに，どのような割り込み処理制御手法が必要となるか考察する．

本研究で評価対象とした環境について述べる．HPC クラウド環境では，Kubernetes を用いたコンテナ環境を用いることが多いものの，本研究では，HPC アプリケーションと割り込み処理による OS ノイズの関係を明らかにするため，HPC アプリケーションのコアを指定して実行可能なベアマシンで実行して評価した．また，HPC アプリケーションを実行する場合，空きコアがある環境と HPC アプリケーションが全 CPU コアを使用する環境がある．評価では，空きコアがある環境において，既存の割り込み制御手法を利用した場合と空きコアにのみ割り込みを許可した場合の GROMACS [4] の処理時間を比較した．また，共存するアプリケーションプログラムとして，iperf3 [5] を実行し，既存の割り込み制御法の特徴を評価し，割り込み処理が集中して実行されるコアが存在する場合の HPC アプリケーションへの影響を評価した．

空きコアがある環境における評価で，GROMACS と iperf3 を同時に実行した環境における GROMACS の処理時間は，GROMACS のみを実行した環境に比べ最大 14% 性能低下することを示した．また，空きコアに割り込みを集中させることにより，アプリケーションの性能が既存の割り込み制御手法を利用した場合に比べ最大 5%性能改善することを示した．これにより，GROMACS と iperf3 を同時に実行した環境では，既存の割り込み制御手法はいずれも割り込みの制御が有効に行われておらず，特定のコアに割り込みが集中する要因の 1 つとなっていることを示した．このような既存の割り込み制御手法の課題を解決し，空きコアがない環境で OS ノイズを削減するような割り込み制御手法の必要性を示した．

2. 技術的背景

2.1 HPC クラウドの概要

大規模計算の需要が高まり，HPC の利用が増加している．しかし，HPC をオンプレミスで導入することは，高コストである．このため，クラウド上で HPC を利用できる HPC クラウドが注目されている．HPC クラウドでは，ユーザがインフラを構築・運用する必要がないため，オンプレミスに比べ，より低コストで HPC を利用できる．

HPC クラウドで想定する環境について述べる．HPC アプリケーションを実行するクラウド環境として，Kubernetes を用いたコンテナ環境を想定する [6]．コンテナは，VM（Virtual Machine）に比べ軽量であり，アプリケーションの可搬性が高い．このため，HPC アプリケーションを実行するクラウド環境に適している．また，クラウド環境では，Ethernet ファブリックで接続したコンテナがマルチテナントで動作している．

2.2 HPC クラウドにおける課題

HPC クラウドで動作するアプリケーションでは，MPI（Message Passing Interface） [7] を用いた並列処理が広く用いられている．MPI は，並列処理で標準的に用いられている通信インタフェースであり，複数の CPU コアがメッセージを送受信することで同期を行う．このため，特定の CPU コアでアプリケーションの処理がシステムソフトウェアの処理で遅延すると，並列処理の同期が遅延し，アプリケーションの性能低下を引き起こす．このようなアプリケーションの性能低下を引き起こすシステムソフトウェアの処理を OS ノイズという．

HPC クラウドが前提としている HPC では，HPC アプリケーションが利用する演算コアとは別に，OS を実行する専用のアシスタントコアを持つ [8] など OS ノイズを考慮した設計になっている．しかし，HPC クラウドでの利用を想定している汎用的なクラウド環境は，OS ノイズを考慮した設計になっていない．このため，汎用的なクラウド環境で HPC アプリケーションの性能を十分発揮するには，OS ノイズを削減することが重要となる．

本研究では，OS ノイズの原因の 1 つとして，通信処理などで発生する割り込み処理に着目する．特定の CPU コア上でアプリケーションの実行中に割り込みが発生すると，アプリケーションの処理を一時中断し，割り込み処理を行う．HPC アプリケーションの場合，特定の CPU コアで割り込みが発生し，当該コアでのアプリケーションの処理完了が遅延すると，並列処理における同期が遅延し，アプリケーション全体の性能低下となる．

割り込み処理に着目したときに，特に課題となることについて述べる．

(1) 特定のアプリケーションの処理に起因する NIC（Network Interface Card）の割り込みが他のアプリケーションを実行中の CPU コアに発生すると，他のアプリケーションの性能に悪影響を与えてしまう．

(2) 複数の CPU コア上で HPC アプリケーションを実行する際，特定の CPU コアに割り込みが集中すると，並列処理の同期が遅延し，アプリケーション全体が性能低下する．

表 1　評価に使用した計算機とスイッチングハブ

計算機（2 台）				スイッチングハブ
CPU	RAM	NIC	カーネル	
Intel Core i7-11700K	16 GB	BlueField-2 (10 Gbps)	Linux Kernel	QNAP QSW-1208-8C
(8 コア，3.60GHz)		X550-T2 (10 Gbps)	5.4.0-89-generic	NETGEAR PROSAFE XS508M

2.3 Linux における割り込み処理の制御手法

OS には割り込み先コアを制御する方法がある．たとえば, Linux には, 割り込み先コアを静的に指定する方法, および動的に制御する手法がある．静的に指定する方法として，Linux では，割り込み先コアを制御する `smp_affinity` [9] がある．`smp_affinity` を静的に設定することで，特定の割り込み要求を処理する割り込み先コアを制御できる．たとえば，この制御手法は，空きコアがある環境で有効である．空きコアで割り込みを処理することができれば，HPC アプリケーションが動作する CPU コアで割り込み処理による OS ノイズは発生しない．しかし，HPC アプリケーションがシステム内の全 CPU コアを使用する環境では，特定の CPU コアに割り込みが集中し，OS ノイズが増加する可能性がある．このため，このような環境では，割り込み先コアを動的に制御する方法が有効である．

割り込み先コアを動的に制御する手法として，Linux には irqbalance [10]，NIC がサポートする機能には RSS（Receive Side Scaling）[11] といった分散処理機構がある．各分散処理機構の概要を以下に示す．

irqbalance： Linux のデーモンプロセスで，10 秒間隔で割り込み状況を確認し，システム全体の性能や消費電力を最適化するように割り込み先コアを制御する．具体的には，10 秒間隔で `smp_affinity` の値を更新し，割り込み先コアを制御する．

RSS： NIC がパケットのヘッダ情報から割り込み先コアを制御し，受信したパケットを複数のコアに分散し処理する．具体的には，5-tuple（送信元先アドレスとポート，プロトコル番号）を確認し，割り込み先コアを制御する．5-tuple が同じパケットは，同じコアに割り込むように制御する．これにより，CPU キャッシュを有効活用でき，TCP reordering による性能低下を抑制できる．

3. 割り込み処理による OS ノイズの影響評価

3.1 評価の目的

評価では，割り込み処理による OS ノイズを削減するための割り込み制御手法の課題を明確化することを目的とする．また，空きコアがある場合に，空きコアを活用する割り込み制御を行った場合の効果についても明らかにすることを目的とする．

割り込み制御手法の課題を明確化するために，RSS や irqbalance といった既存の割り込み制御手法を適用した環境と `smp_affinity` を用いて割り込み先コアを空きコアのみに指定した環境で通信の割り込みによる OS ノイズの影響を評価する．具体的には，各評価環境で GROMACS の処理時間と各 CPU コアの割り込み回数を評価する．これにより，各割り込み制御手法の効果を定量化する．

また，評価結果を基に，HPC クラウドでの利用を想定している汎用的なクラウド環境で HPC アプリケーションの性能を十分発揮するために，OS ノイズを削減するような新しい割り込み制御手法についても議論する．

3.2 評価環境

3.2.1 評価に用いたアプリケーションプログラム

評価に用いたアプリケーションについて述べる．本評価では，割り込み処理のうち，通信処理に伴って発生する割り込み処理に着目した．このため，HPC アプリケーションとして，通信処理が多く，割り込みが多く発生する GROMACS を用いて評価を行った．

GROMACS は, タンパク質, 脂質, 核酸のシミュレーション用に設計された分子動力学のパッケージである．本評価では，GROMACS のベンチマークとして，benchRIB [12] を利用した．また，GROMACS と同時に実行するアプリケーションプログラムとして，通信処理に伴う割り込み処理が多く発生することを想定して，iperf3 を実行した．iperf3 は，ネットワークのパフォーマンスを測定するためのツールである．本測定では，iperf3 の通信帯域は制限せずに実行した．

3.2.2 評価に用いた計算機とネットワーク構成

評価に使用する計算機とスイッチングハブを**表 1** に示す．また，評価時のネットワークと計算機の構成図を**図 1** に示し，コアとアプリケーションの関係図を**図 2** に示す．

Kubernetes を用いたクラウド環境では，現状アプリケーションを実行する CPU コアを固定することができない．このような環境ではプロセスの計算処理と割り込みによる OS ノイズの 2 つについて，双方の性能への影響を区別することが難しく，定量評価の妨げとなる．そこで本評価では，プロセスと割り込みの関係をより明示的に扱うことのできるベアマシンの環境を用いた．ベアマシン環境であれば，アプリケーションのオプションや Linux の taskset コマンドなどを用いることで，図 2 のようにアプリケーションを実行する CPU コアを固定することができる．これに

表 2 評価項目

評価	評価項目	計算機の台数	CPU コアの割り当て方 (各計算機上で同様)	割り込み制御	RSS, irqbalance
1	GROMACS-only (割り込み制御なし)	2 台	GROMACS： コア 0〜5	計算機 1：全コアに割り込みを許可 計算機 2：全コアに割り込みを許可	RSS：有効 irqbalance：有効
2	GROMACS+iperf3 (割り込み制御なし)	2 台	GROMACS：コア 0〜5 iperf3：コア 7	計算機 1：全コアに割り込みを許可 計算機 2：全コアに割り込みを許可	RSS：有効 irqbalance：無効
3	GROMACS+iperf3 (割り込み制御なし)	2 台	GROMACS：コア 0〜5 iperf3：コア 7	計算機 1：全コアに割り込みを許可 計算機 2：全コアに割り込みを許可	RSS：有効 irqbalance：有効
4	GROMACS+iperf3 (割り込み制御あり)	2 台	GROMACS：コア 0〜5 iperf3：コア 7	計算機 1：コア 6 にのみ割り込みを許可 計算機 2：コア 6 にのみ割り込みを許可	—

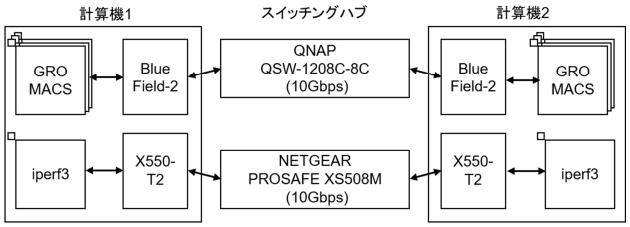

図 1 ネットワークと計算機の構成

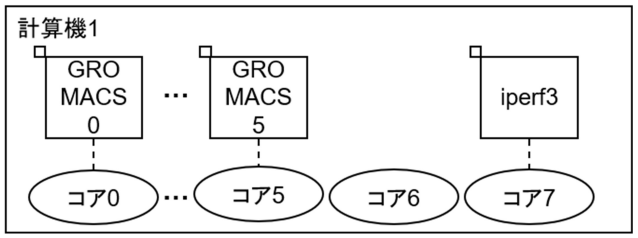

図 2 コアとアプリケーションの関係

より評価のため，空きコアがあるコア割り当て状況の設定や，GROMACS と iperf3 の CPU 処理のコア間分離などが可能となる．

また, GROMACS の通信と iperf3 の通信が経路上の NIC やスイッチングハブで競合すると，割り込み処理回数の低減に伴い OS ノイズの影響が減衰してしまい，定量的な比較が難しくなる．これを避けるため，各アプリケーションの通信する NIC とスイッチングハブはそれぞれ異なる機器を利用した．

具体的には，表 1 で示すように 2 種類の NIC を搭載した計算機を 2 台使用し，GROMACS の通信を BlueField-2，iperf3 の通信を X550-T2 でそれぞれ処理した．BlueField-2 は 10 Gbps 対応スイッチングハブ（QNAP QSW-1208-8C）を介して接続し，X550-T2 は 10 Gbps 対応スイッチングハブ（NETGEAR PROSAFE XS508M）を介して接続した．これにより，割り込み処理の影響評価に際し，通信経路上での通信競合が発生することを回避している．

3.2.3 評価項目

評価項目を**表 2** に示す．空きコアがある環境において，次に示す 4 種類の条件で GROMACS の処理時間と割り込み回数を評価した．

(評価 1)GROMACS のみを実行，既存割り込み制御手法（RSS 有効，irqbalance 有効）

(評価 2)GROMACS と iperf3 を同時実行，既存割り込み制御手法（RSS 有効，irqbalance 無効）

(評価 3)GROMACS と iperf3 を同時実行，既存割り込み制御手法（RSS 有効，irqbalance 有効）

(評価 4)GROMACS と iperf3 を同時実行，空きコアにのみ割り込み許可

評価 1 が GROMACS 単体でのシングルテナント環境を想定しているのに対し，評価 2 から評価 4 は GROMACS と iperf3 の同時実行によるマルチテナント環境を想定した環境となっている．評価 1 と評価 3 を比較することで，通信の割り込みによる OS ノイズが既存の割り込み制御手法を適用した環境においてどのような影響を及ぼしているかを測ることができる．また評価 3 と評価 4 を比較することで，空きコアがある環境における割り込み制御の効果を定量化することができる．評価 2 は評価 3 と併せて，評価環境上での RSS および irqbalance の割り込み制御状況の調査およびその効果の定量化を意図している．

3.3 評価結果と考察

各評価条件で 3 回ずつ GROMACS を実行した．評価結果を**表 3** に示し，GROMACS 実行時の割り込みの分布を**図 3**，**図 4**，**図 5**，および**図 6** に示す．図 3 は評価 1 の割り込みの分布，図 4 は評価 2 の割り込みの分布，図 5 は評価 3 の分布，図 6 は評価 4 の割り込みの分布である．

表 3 より，評価 3 における GROMACS の処理時間は，評価 1 に比べ最大 14%長い．これは，iperf3 の通信により発生する割り込み処理が GROMACS が動作するコア上に発生しているためである．図 3 と図 5 より，評価 3 における各 CPU コアへの割り込み回数は，評価 1 に比べ増加している．つまり，CPU コアをプロセス毎に排他的に利用した場合であっても，特定のプロセスの通信により発生する割り込み処理が他のプロセスが利用している CPU コアに発生し，悪影響を与えている．特に，GROMACS が動作するコア 1 に割り込みが集中し，GROMACS の CPU コア当たりの割り込みの最大値と最小値の差が最大 60 倍と

表 3 GROMACS の処理時間と割り込み回数

通番	評価	評価条件	処理時間 [s]	割り込み回数 (コア 0〜5) [回] 最大値-最小値	割り込み回数 (コア 0〜5) [回] 平均値	割り込み回数 (コア 7) [回]	割り込み回数 (全コア) [回]
1	評価 1	GROMACS-only	2403.825	918,768	4,440,417	2,765,646	30,602,250
2		RSS：有効	2378.567	1,415,829	4,535,404	2,133,938	31,479,581
3		irqbalance：有効	2381.585	2,035,114	4,495,954	917,986	29,219,102
4	評価 2	GROMACS+iperf3	2707.238	2,119,067	7,940,774	38,667,446	91,744,145
5		RSS：有効	2707.275	2,673,754	8,106.200	41,929,582	94,549,350
6		irqbalance：無効	2694.016	3,935,052	7,905,036	41,338,343	93,309,553
7	評価 3	GROMACS+iperf3	2767.992	55,599,510	16,416,056	7,138,894	113,119,755
8		RSS：有効	2790.963	54,402,139	16,599,974	7,797,611	113,429,745
9		irqbalance：有効	2759.848	42,761,059	14,461,883	7,035,780	99,767,067
10	評価 4	GROMACS+iperf3	2644.291	0	0	0	89,952,515
11		コア 6 にのみ	2642.979	0	0	0	94,076,047
12		割り込みを許可	2651.267	0	0	0	95,171,679

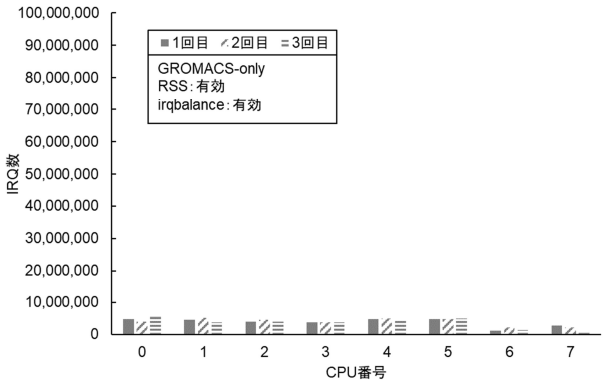

図 3 評価 1 の割り込みの分布

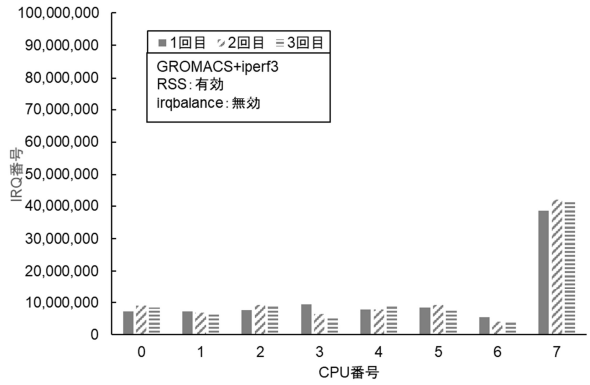

図 4 評価 2 の割り込みの分布

なっている．このような割り込みの特定コアへの集中が並列処理における同期の遅延を引き起こし，GROMACS の処理時間を増加させたことがわかる．

また，表 3 より，空きコアにのみ割り込みを許可する場合の GROMACS の処理時間は，既存の割り込み制御手法を利用した場合に比べ最大 5%短い．これは，空きコアにのみ割り込みを許可することで他のコアへの割り込みがなくなり，OS ノイズによる処理時間の増加が回避されているためである．図 6 より，GROMACS と iperf3 の通信により発生する割り込み処理はすべて空きコアであるコア 6 にのみ発生する．つまり，GROMACS が動作するコアに割り込みが発生しないため，割り込み処理による同期の遅延が発生しない．このように，空きコアがある環境では割り込みをその空きコアに集中させることで HPC アプリケーションの性能を改善できると言える．

評価 2 と評価 3 のように GROMACS と iperf3 を同時に実行した環境において，既存の割り込み制御手法はいずれも割り込みの制御が有効に行われていない．図 4 は既存の割り込み制御手法として RSS のみを，また図 5 は RSS と irqbalance の双方を用いた際の，各 CPU コアへの割り込み回数の分布を示している．図 4 ではコア 7 に，また図 5 ではコア 1 にそれぞれ割り込みが極端に集中していることがわかる．RSS はパケットの 5-tuple を参照して割り込み先コアを制御するため，特定の相手との通信が集中して行われる iperf3 の通信による割り込みに対しては，割り込みの分散として機能していないと言える．また irqbalance は割り込み先コアの変更を行っていることがわかるが，やはり割り込みの分散としては有効に機能していない．このように，既存の割り込み制御手法は GROMACS と iperf3 を同時に実行した環境における割り込みの集中の一因となっている．

`smp_affinity` を用いて静的に割り込み先コアを指定する方法は，空きコアがある環境では有効であるが，空きコアがない環境では特定の CPU コアに割り込みが集中し，OS ノイズが増加する可能性がある．また，RSS や irqbalance といった既存の割り込み制御手法は有効に機能していない．このため，OS ノイズを削減し，汎用的なクラウドのインフラでの HPC 環境実現には，より有効な割り込み制御手法について検討する必要がある．

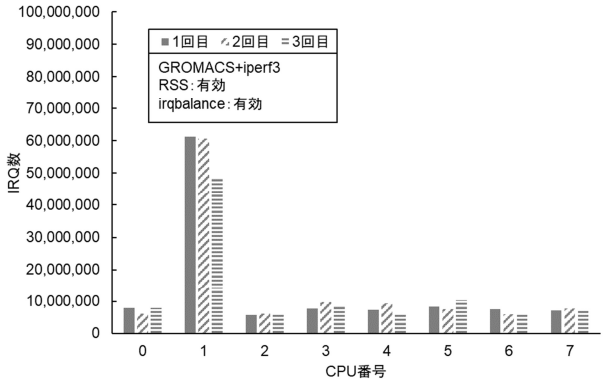

図 5 評価 3 の割り込みの分布

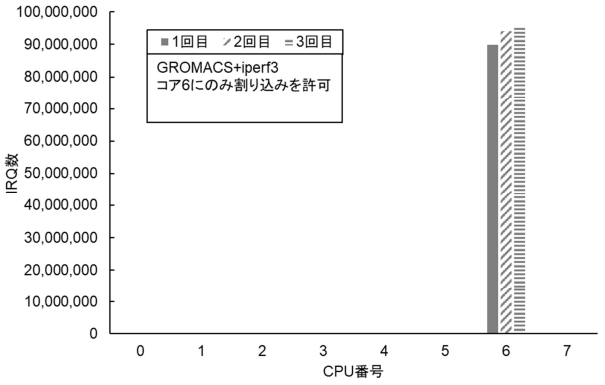

図 6 評価 4 の割り込みの分布

3.4 割り込み処理の制御法についての考察

本評価では，NIC の割り込み処理に着目し，既存割り込み制御手法で評価を行い，割り込みを HPC アプリケーションの実行状態や利用者の意図に合わせて制御するための以下の課題を明らかにした．

(1) パケットのヘッダ情報だけみて制御する手法 RSS では，特定のコアに割り込みが集中することがあるため，通信内容や受信プロセスの情報まで考慮する必要がある．

(2) 一定周期で割り込み状況を確認し，システム全体の性能などを最適化する制御法 irqbalance では，HPC アプリケーション向けに割り込み先コアを制御できていないことを示した．なお，今回は irqbalance の周期は 10 秒で評価したが，周期を短くしたときに，irqbalance の割り込み分散の効果と制御によるオーバヘッドや OS ノイズの影響を評価する必要がある．

(3) コア毎の CPU 負荷，HPC アプリケーションの走行情報などの OS レベルの情報を把握して，活用した割り込み制御手法が必要である．たとえば，HPC アプリケーションを実行していないコアがあった場合に，積極的にそのようなコアを割り込み処理を実行させて活用させたいが，irqbalance では活用できていない．

上記の課題を解決する制御法を実現するために，HPC アプリケーション向け割り込み制御法として，以下のような性質を持つ割り込み制御手法について今後検討する．

(1) HPC アプリケーションを活用していないコアがある場合に，HPC アプリケーションへの OS ノイズ発生を抑制するために，空きコアを活用する割り込み制御手法

(2) 複数の HPC アプリケーションが動作しているときに，他の HPC アプリケーションの OS ノイズの影響を抑制する割り込み制御手法

今後は，これらの性質を満たす割り込み制御手法を検討し，OS ノイズを削減できる HPC クラウド実行基盤を検討する．

4. 関連研究

最新のスーパコンピュータである富岳 [13] では，A64FX という専用の CPU を搭載している．A64FX は，アプリケーションが使用する演算コアとは別にアシスタントコアを持つ [8]．このアシスタントコアで OS ノイズの原因となる IRQ やカーネルスレッドの処理をアシスタントコアで実行することで，OS ノイズを削減している．本研究では，汎用的なクラウドのインフラで HPC クラウドを実現した際に，割り込み処理による OS ノイズを削減する制御法を実現することであるため，アプローチが異なる．

また，文献 [14] と文献 [15] は，クラウド環境で HPC アプリケーションの性能を評価している．文献 [14] では，HPC とクラウドコンピューティングの間の性能差を調査するために，現在の HPC とクラウドの性能差を調査している．調査の結果，クラウドは，計算インテンシブのワークロードで HPC に匹敵すること，1 対 1 通信において，HPC より高い帯域幅と低レイテンシを提供することを述べている．また，クラウドは，メモリインテンシブのワークロードで，計算インテンシブのワークロードに最適化されたクラウドでは，性能が低下すること，およびメモリインテンシブのワークロードに最適化されたクラウドでは，HPC を上回る性能であることが述べられている．文献 [15] では，汎用的なクラウド環境で HPC アプリケーションを実行する方法として，共有メモリを用いることを述べている．多くのクラウド環境では，ネットワークに TCP プロトコルを用いているが，TCP にはレイテンシが高いという問題がある．このため，文献 [15] では，TCP の代わりに共有メモリを用いて効率的に通信することで，クラウド環境においても HPC 環境と同等の性能を発揮することを述べている．本研究は，割り込み処理による OS ノイズや割り込み制御手法について評価，検討している点で異なる．

文献 [16] は，クラウド環境とオンプレミスな HPC 環境で HPC アプリケーションを実行した際のネットワーク性能と OS ノイズが大規模計算のスケーラビリティに与える

影響を分析している．分析結果より，オンプレミスな HPC 環境では，OS ノイズの影響は小さいが，クラウド環境では OS ノイズの影響が大きいと述べている．本研究は，OS ノイズの中でも割り込み処理による OS ノイズに着目し，OS ノイズがアプリケーションに与える影響を評価しており，観点が異なる．

5. おわりに

HPC クラウドでの利用を想定している汎用的なクラウド環境は，OS ノイズを考慮した設計になっていない．このため，汎用的なクラウド環境で HPC アプリケーションの性能を十分発揮するには OS ノイズを削減する必要がある．本稿では，OS ノイズの原因の 1 つである割り込み処理に着目し，OS ノイズがアプリケーションに与える影響を評価した．

HPC アプリケーションを実行する場合，空きコアがある環境と HPC アプリケーションが全 CPU コアを使用する環境がある．今回は，空きコアがある環境で，割り込み処理による OS ノイズの影響を評価した．評価結果より，割り込みを空きコアに集中させることにより，アプリケーションの性能が既存の割り込み制御手法を利用した場合に比べ，最大 5%改善することを示した．また，GROMACS と iperf3 を同時に実行した環境において，既存の割り込み手法はいずれも割り込みの制御が有効に行われていないことを示した．

既存の割り込み制御手法を用いた評価により，割り込みを HPC アプリケーションの実行状態や利用者の意図に合わせて制御するための課題を明確化した．今後は，この課題を解決するような HPC アプリケーション向けの割り込み制御手法を検討し，OS ノイズを削減できる HPC クラウド実行基盤を検討する．

参考文献

[1] AWS: ハイパフォーマンスコンピューティングサービス, 入手先 ⟨https://aws.amazon.com/jp/hpc/⟩ (参照：2023-01-29).
[2] Azure: ハイパフォーマンスコンピューティング, 入手先 ⟨https://azure.microsoft.com/ja-jp/solutions/high-performance-computing/⟩ (参照：2023-01-29).
[3] GCP: ハイパフォーマンスコンピューティング, 入手先 ⟨https://cloud.google.com/solutions/hpc/⟩ (参照：2023-01-29).
[4] GROMACS development team: GROMACS, 入手先 ⟨https://www.gromacs.org/⟩ (参照：2023-01-29).
[5] iPerf - The TCP, UDP and SCTP network bandwidth measurement tool, 入手先 ⟨https://iperf.fr/⟩ (参照：2023-01-29).
[6] 楠 恒輝，加藤 純，佐藤 充：HPC クラウドでの OS ノイズの定量評価，コンピュータシステム・シンポジウム論文集，Vol.2022，pp.56–62 (2022).
[7] mpi forum: MPI Forum, 入手先 ⟨https://www.mpi-forum.org/⟩ (参照：2023-01-29).
[8] Gerofi, B., Tarumizu, K., Zhang, L., Okamoto, T., Takagi, M., Sumimoto, S. and Ishikawa, Y.: Linux vs.Lightweight Multi-Kernels for High Performance Computing: Experiences at Pre-Exasc, *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC'21)*, pp.1–13 (2021).
[9] The Linux Kernel: SMP IRQ affinity, 入手先 ⟨https://docs.kernel.org/core-api/irq/irq-affinity.html⟩ (参照：2023-01-30).
[10] Irqbalance: Irqbalance, 入手先 ⟨https://github.com/Irqbalance/irqbalance⟩ (参照：2023-01-30).
[11] Intel: RSS (Receive-Side Scaling) , 入手先 ⟨https://www.intel.co.jp/content/dam/support/us/en/documents/network/sb/318483001us2.pdf⟩ (参照：2023-01-30).
[12] MAX-PlANCK-INSTITUTE: benchRIB, 入手先 ⟨https://www.mpinat.mpg.de/grubmueller/bench⟩ (参照：2023-01-29).
[13] Fujitsu: スーパーコンピュータ「富岳」, 入手先 https://www.fujitsu.com/jp/about/businesspolicy/tech/fugaku/ (参照：2023-02-01).
[14] Guidi, G., Ellis, M., Buluç, A., Yelick, K., and Culler, D.: 10 Years Later: Cloud Computing is Closing the Performance Gap, *In Companion of the ACM/SPEC International Conference on Performance Engineering (ICPE '21)*, pp.41–48 (2021).
[15] Walkup, R., Seelam, S. R. and Wen, S.: Best Practices for HPC Workloads on Public Cloud Platforms: A Guide for Computational Scientists to Use Public Cloud for HPC Workloads, *Proceedings of the 2022 ACM/SPEC on International Conference on Performance Engineering (ICPE'22)*, pp.29–35 (2022).
[16] De Sensi, D., De Matteis, T., Taranov, K., Di Girolamo, S., Rahn, T., and Hoefler, T.: Noise in the Clouds: Influence of Network Performance Variability on Application Scalability, *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, Vol. 6, pp.1–27 (2022).