Azure CycleCloud 利用環境と仮想マシン上でのベンチマークテスト結果

永井 亨 [1)], 五十木 秀一 [2)], 河合 直聡 [1)], 片桐 孝洋 [1)], 星野 哲也 [1)]

1) 名古屋大学 情報基盤センター
2) 日本マイクロソフト株式会社

nagai@nagoya-u.jp

Usage environment of Azure CycleCloud and benchmark test results on VMs

Toru Nagai [1)], Shuichi Gojuki [2)], Masatoshi Kawai [1)], Takahiro Katagiri [1)], Tetsuya Hoshino [1)]

1) Information Technology Center, Nagoya Univ.
2) Microsoft Japan Co., Ltd.

概要

パフリッククラウドの利用環境の調査を主な目的として Microsoft Azure を対象にした仮想マシンの性能測定を行った．具体的には名古屋大学情報基盤センターと日本マイクロソフト社との共同研究のもとで HPC 利用環境に特化した Azure CycleCloud を使用して種々のベンチマークプログラムを仮想マシン上で実行した．本稿では Azure CycleCloud の利用環境と仮想マシン上でのベンチマークテスト結果について報告し，スーパーコンピュータシステムとパブリッククラウドの連携について考察する．

1 はじめに

名古屋大学情報基盤センター（以下，名大センター）は，多様化する利用者の要求に応える計算機システムを目指して異なる特徴をもつ複数のサブシステムと複数のストレージシステムで構成されるスーパーコンピュータ「不老」のサービスを 2020 年 7 月に開始した[1]. サブシステムの 1 つである「クラウドシステム」はオンプレミス型クラウドの機能をもち，研究室サーバでは処理能力が不足する研究者の計算需要に応える役割を担っている. しかし,「クラウドシステム」の稼働率は 2020 年度 81.2%，2021 年度 72.0%，2022 年度 75.2%，2023 年度 76.4%と高稼働率を維持している. また，稼働率の季節的変化も小さいため定常的にターンアラウンドタイムが長くなっている．計算負荷を分散しターンアラウンドタイム増大の問題を解決する方策の 1 つとして，オンプレミス計算資源の処理能力が閾値を超えた場合にパブリッククラウドを利用するクラウドバースト[2]が考えられる．

また，キャンパス内の消費電力量を抑制するため 2022 年 6 月以降 Type I サブシステムを，2023 年 4 月から Type II サブシステムをそれぞれ縮退運転している．このため大規模ジョブは実行されにくい状況が続いている．特に，サブシステム全系を使用するような超大規模ジョブは事実上実行できない. このような状況において大規模ジョブ，超大規模ジョブの利用者への影響を緩和する方策としてパブリッククラウドの利用が有効であると考えられる．

一方，筆者の中で名大センターに所属してスパコン運用に携わる者はそもそもパブリッククラウドの利用経験が少ない．そこで，我々はパブリッククラウド利用環境の調査を目的として日本マイクロソフト社との共同研究のもと， Microsoft Azure を対象に仮想マシンの性能測定おこなった．具体的には 2023 年 3 月末から 6 月末にかけて HPC 利用環境に特化した Azure CycleCloud[3]を使用して仮想マシン上で種々のベンチマークプログラムの性能測定を行った．

以下では，2 章で Azure CycleCloud の概要を述べ，3 章でベンチマークテストの測定結果を報告する．4 章でスーパーコンピュータシステムとパブリッククラウドとの連携について考察し，最後にまとめを述べる．

2 Azure CycleCloud の概要

Azure CycleCloud は Microsoft Azure で提供されるサービス群の 1 つで HPC 環境を構築し管理するためのツールである．CycleCloud の機能は，必要なリソースをデプロイして HPC 環境を構築する機能と構築したシステムを管理する機能の 2 つに

分けられる．

(1)　HPC 環境構築機能

CycleCloud は仮想マシンやストレージをオーケストレーションしてジョブスケジューラ設定済みのクラスタを既存のネットワークにデプロイする．一般的に使用されるジョブスケジューラ（PBS Professional OSS，Slurm，IBM LSF，Grid Engine，HT Condor）は CycleCloud に組み込まれている．仮想マシンに CPU，主記憶，GPU，ネットワーク帯域幅などの資源を割りあてたものを VM サイズとよぶ．CycleCloud はこの VM サイズに基づいて仮想マシンをデプロイする．例えば，VM サイズ HB120rs_v3 は CPU が AMD EPYC 7V73X (Milan-X)×2 ソケット，コア数 120，L3 キャッシュサイズ 768MB×2 ソケット，主記憶 448GiB，200Gbps InfiniBand HDR を意味する．表 1 に使用した VM サイズを示す．仮想マシンの OS は Linux と Windows が選択できる．ストレージには Azure ネイティブな HPC 向け NFS である Azure NetApp Files のほかに Lustre，BeeGFS などが選択できる．Azure NetApp Files では Standard，Premium，Ultra の 3 つのサービスレベルがある．Standard, Premium, Ultra のスループット性能は TiB あたりの性能が決められており，4TiB 使用するとき，それぞれ，64MiB/s，256MiB/s，512MiB/s である．

(2)　システム管理機能

CycleCloud 管理者に構築した HPC 環境下の利用者管理機能，資源管理機能，課金管理機能を提供する．また，ジョブ処理のワークフローをオーケストレーションする機能を提供する．

表 1　VM サイズ

VM サイズ	CPU	ソケット数	物理コア数	HW スレッド数	L3 キャッシュサイズ	メモリ容量 (GiB)	Inter-connect
HB120-16rs_v3	AMD EPYC 7V73X (Milan-X)	2	16	16	768MB x 2 sockets	448	IB HDR 200Gbps
HB120rs_v3	AMD EPYC 7V73X (Milan-X)	2	120	120	768MB x 2 sockets	448	IB HDR 200Gbps
HC44rs	Intel Xeon Platinum 8168 (Sky Lake)	2	44	44	33MB x 2 sockets	352	IB EDR 100Gbps
D96s_v5	Intel Xeon Platinum 8370C (Ice Lake)	2	48	96	48MB x 2 sockets	384	―

CycleCloud を利用して構築した HPC 環境の概要を図 1 に示す．実線で囲った部分が Microsoft Azure である．データセンターは世界各地にある．我々はリージョン米国東部を選択した．図中の CycleCloud サーバはシステム管理を担う．サーバおよびノードの下のカッコ内は VM サイズをあらわす．CycleCloud を利用したジョブ実行の流れは以下のようになる．

1. 利用者はローカル端末からマスターノードに公開鍵認証による SSH 接続をおこなう．
2. マスターノード上でジョブスクリプトを作成しスケジューラにジョブを投入する．
3. CycleCloud はスケジューラと連携して要求された計算資源を自動的にデプロイする．このとき，需要に応じて計算資源のノード数を自動的に拡張することができる．
4. ジョブを実行する．
5. ジョブが終了し次のジョブがない場合，計算資源は自動的に停止する．

したがって，図 1 に示した計算ノード群（仮想マシン）はジョブが投入されるまで待機しているわけではなく，資源要求があったときにデプロイされ，ジョブが終了すると解放される．常時稼働する Azure 側の資源は CycleCloud サーバ，マスターノード，Azure NetApp Files である．繰り返しになるが，2 から 4 の手順に示したようにジョブ投入後ただちにジョブが実行されるわけではなく，計算資源がデプロイされたのちジョブ実行が開始される．ジョブ投入から実行開始までの時間はクラウド全体の混雑状況にも依存するが，通常は 5 分程度であった．

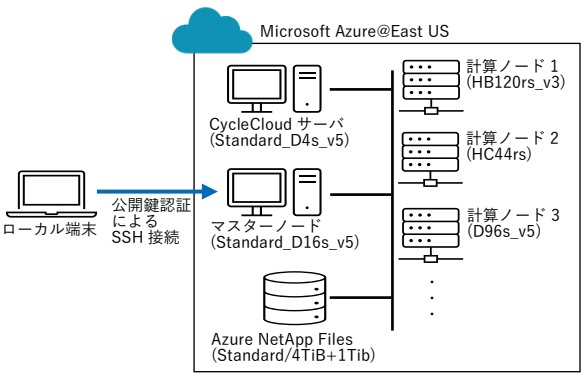

図 1　Azure CycleCloud の概要

利用したソフトウェアは以下のとおりである．

- OS：CentOS 7.9
- スケジューラ：PBS Professional OSS
- コンパイラ/開発環境 : GCC 9.2.0，Intel oneAPI 2021.2.0，Intel oneAPI 2023.0.0，NVIDIA HPC SDK 22.11
- MPI ライブラリ：Intel MPI 2021.4.0，MVAPICH2.3.6，OpenMPI 4.1.1，HPC-X v2.8.3

これらの中で Intel oneAPI 2021.2.0，Intel oneAPI 2023.0.0，NVIDIA HPC SDK 22.11 は環境構築時に別途インストールした．それ以外は標準で備わっ

ているものである．また，ストレージは Azure NetApp Files（Standard，4TiB）を使用した．

3 ベンチマークプログラムの実行結果

3.1 STREAM Triad

STREAM ベンチプログラム Triad[4]の測定ではコンパイル時のコンパイラとコンパイラオプション，実行時環境変数の組み合わせを変えて測定した．コンパイラ，コンパイラオプション，環境変数の組み合わせを付録中の付表 A に示す．組み合わせを表す実行形態の表記形式が付表 A の一番左側のコラムに示されている．例えば，gcc 9.2.0 でオプション-march=native -O3 -fopenmp を指定してプログラムをコンパイルし，環境変数 OMP_PLACES=threads，OMP_PROC_BIND=close として実行モジュールを実行したとき，gcc9.2.0+threads+close と表記する．

STREAM ベンチマークプログラムの冒頭には配列サイズをキャッシュサイズの 4 倍以上にとる必要があると記述されている．表 1 に示したように HBv3 系列の VM サイズでは 2 ソケット合計で 1.536GB の L3 キャッシュをもつ．このため配列サイズを$N = 1 \times 10^9$（8GB）とした．他の VM サイズでも同じ配列サイズを指定した．

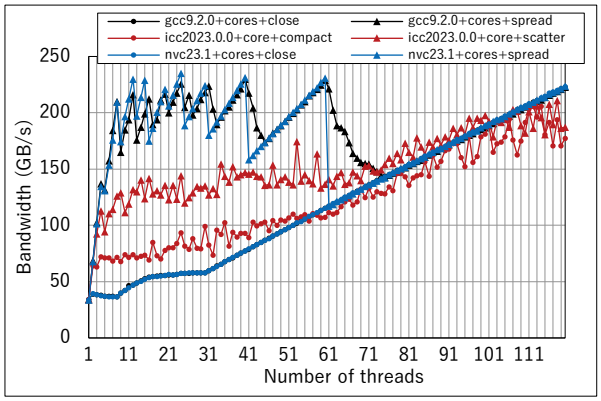

図 2　STREAM Triad の結果（HB120rs_v3）

VM サイズ HB120rs_v3（AMD EPYC 7V73X (Milan-X)×2 ソケット，ソケットあたり 60 コア）の仮想マシン上でスレッド数を 1 から 120 まで変化させたときの結果を図 2 に示す．図中の凡例は付表 A の表記形式にしたがう．コンパイラ，コンパイラオプション，実行時の環境変数の組み合わせで結果が明らかに異なる．メモリ帯域が最大になったのは nvc23.1+cores+spread, スレッド数 24 のときの 235.1GB/s であった. また, VM サイズ HBv3 系列中の最大値は HB120-16rs_v3 仮想マシン上の icc2023.0.0+core+compact，スレッド数 16 のときの 248.1GB/s であった．Azure web サイト[5]の STREAM ベンチマーク結果（配列サイズ$N = 4 \times 10^8$）では，HBv3 系列の最大メモリ帯域は 358GB/s と報告している．

VM サイズ HC44rs（Intel Xeon Platinum 8168 (Sky Lake)×2 ソケット，ソケットあたり 22 コア）の仮想マシン上でスレッド数を 1 から 44 まで変化させたときの結果を図 3 に示す．Intel コンパイラではオプション-qopt-zmm-usage=high（512 ビット長の zmm レジスタを利用して SIMD 化を促進する）を指定した場合とそうでない場合とが示されている．詳細は不明だがメモリ帯域を測定するプログラムでもこのオプションが有効となる場合があることを示している．最大メモリ帯域は icc2023.0.0+zmm_high+core+scatter，スレッド数 32 で 203.0GB/s であった．

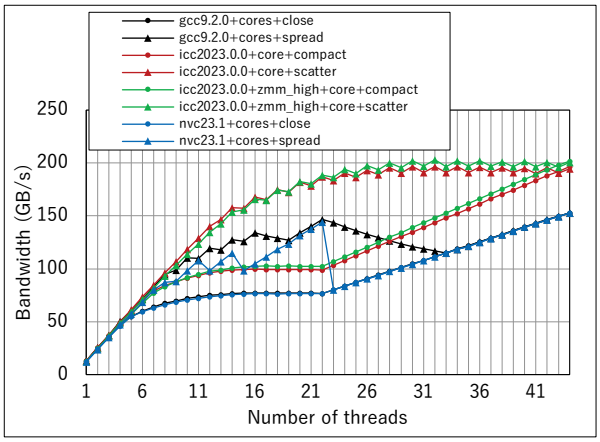

図 3　STREAM Triad の結果（HC44rs）

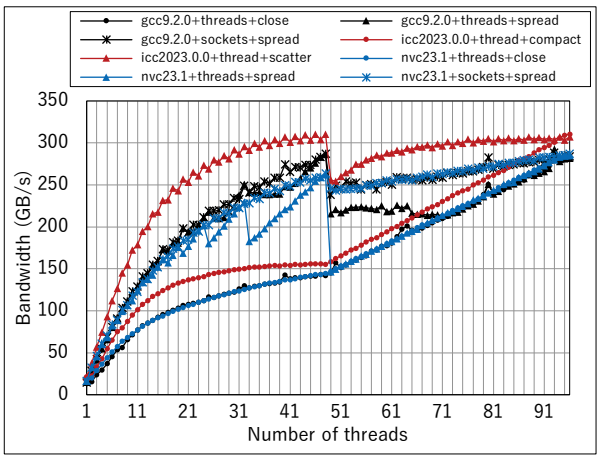

図 4　STREAM Triad の結果（D96s_v5）

VM サイズ D96s_v5（Intel Xeon Platinum 8370C (Ice Lake)×2 ソケット，ソケットあたり 24 コア，ソケットあたり 48 ハードウェアスレッド）の仮想マシン上でスレッド数を 1 から 96 まで変化させたときの結果を図 4 に示す．VM サイズ D96s_v5

の CPU である Intel Xeon Platinum 8370C (Ice Lake) はコアあたり 2 つのハードウェアスレッドを備えている．このため環境変数 OMP_PLACES=threads の場合と OMP_PLACES=cores の場合を比較してみたが顕著な差異はみられなかった．図には OMP_PLACES=threads としたときの結果が示されている．また，図 4 に示すように gcc 9.2.0 と nvc 23.1 では OMP_PLACES=sockets とした方が全般的によい結果がえられた．icc2023.0.0+thread+scatter，スレッド数 48 で最大メモリ帯域 310.4GB/s が得られた．

3.2 OSU Micro-Benchmarks

OSU Micro-Benchmarks [6]のバージョンは 7.1-1 である． HB120-16rs_v3 仮想マシン上で実行した MPI レイテンシの測定結果を図 5 に，MPI バンド幅の測定結果を図 6 にそれぞれ示す．図 5，6 には 8 種類のコンパイラと MPI ライブラリの組み合わせがプロットされている． HB120-16rs_v3 の MPI レイテンシは 1.9～4.3μs，MPI バンド幅の最大値は 24.7GB/s であった

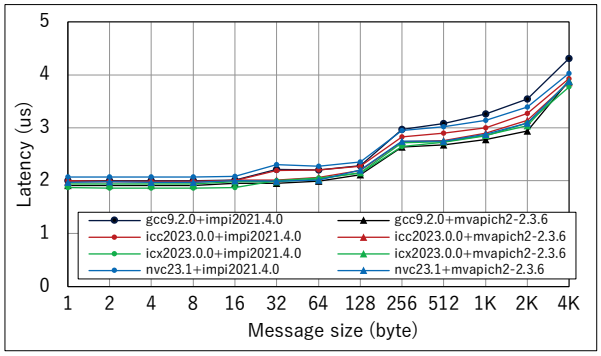

図 5　MPI レイテンシ（HB120-16rs_v3）

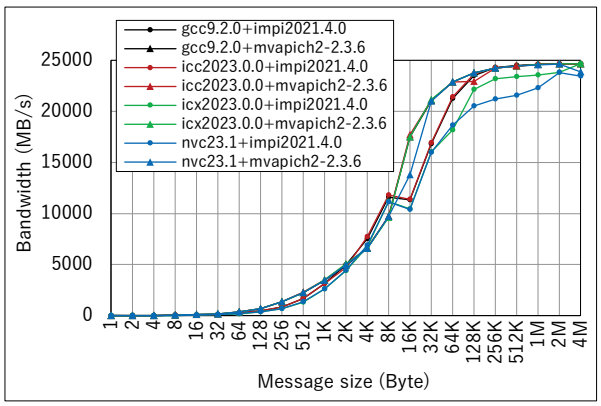

図 6　MPI バンド幅（HB120-16rs_v3）

なお，仮想マシンはジョブ実行の都度デプロイされるため，実行毎にネットワーク的な距離が変化するためレイテンシも変化することが予想される．本研究では 1 つのジョブの中でコンパイラと MPI ライブラリが異なる複数の組み合わせの実行モジュールを実行しているので図 5 に示したレイテンシの差異は有意であると考える．また，バンド幅は上記のネットワーク的距離に加えて Azure クラウドの他の利用者の利用状況の影響を受ける可能性があることにも留意する必要がある．

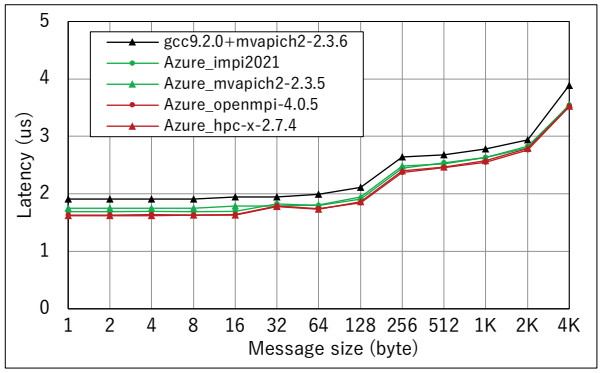

図 7　MPI レイテンシの比較（HBv3 系列）

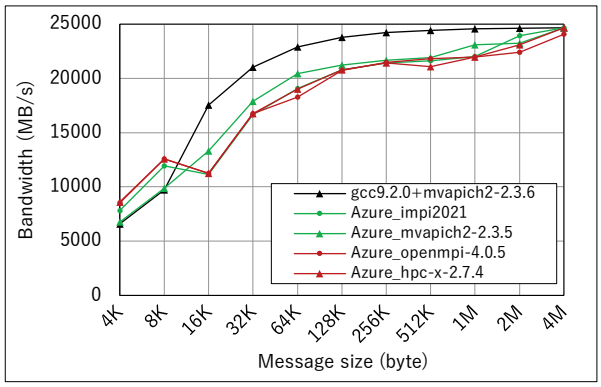

図 8　MPI バンド幅の比較（HBv3 系列）

Azure web サイト[7]では HBv3 系列における OSU Micro-Benchmarks（バージョンは 5.7）の結果が公開されている．比較のため，本研究での結果と web サイトに掲載された結果を重ねて表示したものを図 7 および図 8 に示す．図 7 と図 8 の凡例で先頭に Azure を付したものが web サイトの結果である．web サイトには HBv3 系列のどの VM サイズで測定したか，また，どのコンパイラを使用したかの記載はいずれもなかった．図 7 をみるとレイテンシでは web サイトの方が 8～15%ほどよい値が出ている．一方，図 8 をみるとバンド幅では逆に本研究の方がよい結果となった．図 7 と図 8 ではメッセージサイズの範囲が異なるから互いに矛盾する結果というわけではない．

図 9，10 に VM サイズ HC44r の場合の MPI レイテンシと MPI バンド幅をそれぞれ示す．HC44r 仮想マシンの MPI レイテンシは 1.5～3.7μs，最大 MPI バンド幅は 11.9GB/s であった．したがって，HB120-16rs_v3（IB HDR 200Gbps）と比較すると，MPI バンド幅は半分以下であるが，MPI レイテン

シは優位である．

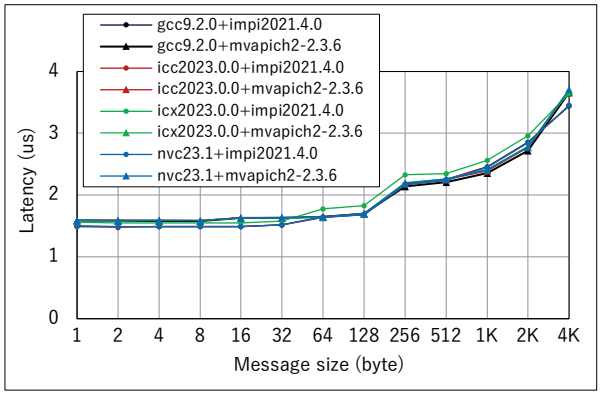

図 9　MPI レイテンシ（HC44rs）

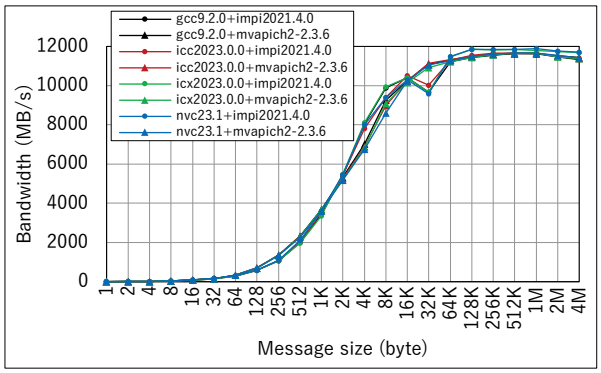

図 10　MPI バンド幅（HC44rs）

3.3 反復法ソルバ

次に，実用的なアプリケーションとして連立一次方程式ソルバの性能測定を行った．選択した連立一次方程式ソルバはブロック不完全コレスキー分解前処理(BIC)つきクリロフ部分空間(CG)法である．BICCG では以下の方程式の求解を目的とする．

$$Ax = b, \qquad A \in \mathbb{R}^{n\times n}, \qquad x, b \in \mathbb{R}^{n}$$

本方程式において，Aを係数行列，xを解ベクトル，bを右辺ベクトルとする．BICCG 法では，以下のプロセスに基づいて近似解x^kを得る．ここで，BIC 前処理で使用するU，D は行列Aからブロック不完全コレスキー分解によって得られる行列であり，図 11 に示した主計算前に実施する．また，BIC 前処理は，逆行列を算出せず，前進後退代入で実施する．BICCG ソルバは MPI/OpenMP ハイブリッド並列化が実施されている．なお，BIC 前処理の前進後退代入は一般的に計算依存性があるが，多色順序付けによって並列化を実施している[8].

測定では，ポアソン方程式を有限差分法によって離散化した問題および Florida Sparse Matrix Collection より入手した G3 Circuit の求解を実施した．ポアソンの問題サイズは 256×256×512=33M，G3 Circuit は 1.5M である．また，実行ではノードあたり 2 プロセスとしている．それ以外の測定条件を表 2 に示す．

do k = 1, n

$$\alpha = \frac{(r^k,\ p)}{(p,\ Ap)}$$

$x^k = x^{k-1} + \alpha p$　　p:探索ベクトル

$r^k = r^{k-1} - \alpha Ap$　　r:残差ベクトル

if (r^k<= 閾値) exit (計算終了)

$q^k = U^{-1}D^{-1}U^{-t}r^k$　　BIC 前処理

$$p = q^k + \frac{(q^k, r^k)}{(q^{k-1}, r^{k-1})}p$$

enddo

図 11　BICCG 法のアルゴリズム

表 2　測定条件一覧

コンパイル条件	コンパイラ	Intel (2021.2.0)
	コンパイラオプション	-O3 -xHost
ソルバパラメータ	相対残差閾値	1.0×10^{-7}
	ブロックサイズ	32
	色数	20

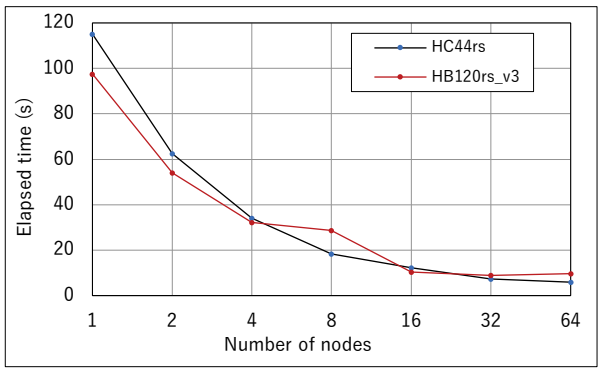

図 12　ポアソン問題の測定結果

図 12 に Poisson 問題の VM サイズ HC44rs および HB120rs_v3 仮想マシンでの測定結果を示す．いずれの仮想マシンでもある程度のスケーラビリティを確認できた．また，HC44rs と HB120rs_v3 の比較では，ポアソン問題の並列度が小さい範囲では HB120rs_v3 のほうが早い傾向が見て取れる．これは，図 2，3 に示した Triad の性能差が反映されたこと，および，HB120rs_v3 のコア数が大きいためと推測する．対して並列度が大きい範囲では HC44rs が早い傾向を示している．これは並列度が大きい範囲では，計算時間が十分に短くなっており，通信時間が見えているためである．本アプリのメッセージサイズは小さく，レイテンシセンシティブである．3.2 節で論じたように MPI レイテ

ンシでは HB120rs より HC44rs の方が優位であるため上述のような差が出たと推定する．

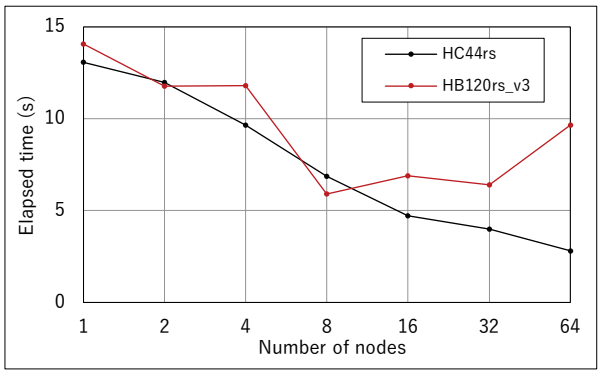

図 13　G3 Circuit の測定結果

図 13 に G3 Circuit の測定結果を示す．HC44rs ではスケーラビリティが確認できるが，HB120rs_v3 ではばらつきが大きい．G3 Circuit では自由度が小さいためそもそも通信が支配的である．このためノード数を増やすとデプロイされたノード間のネットワークホップ数の影響が大きくなり，通信時間のばらつきが大きくなったと推定する．

4 考察

ここでは Azure CycleCloud の利用経験をもとにスーパーコンピュータシステムとパブリッククラウドの連携について考える．先行研究として大阪大学サイバーメディアセンター（以下，阪大センター）の伊達らによる阪大センター大型計算機システム OCTOPUS[9]と Microsoft Azure とを連携させる OCTOPUS-Azure クラウドバースティング実証実験がある[10,11]．さらに，文献[12]では OCTOPUS より遅れて稼働を開始したシステム SQUID[13]と Azure を連携させた SQUID-Azure クラウドバースティング環境を構築し，構築した環境が具備すべき機能，性能，価格について検討されている．

以下では単純化のため，x86 アーキテクチャの CPU と主記憶からなるノードが Infiniband で結合されたクラスタと共有ストレージで構成されるシステムについて，Case 1：クラウドバーストの場合と Case 2：利用者が積極的にパブリッククラウドを利用するの場合の 2 つを考える．

Case 1：クラウドバーストの場合

この場合には利用者は投入したジョブがオンプレミス環境で実行されているか，パブリッククラウドで実行されているかを意識することなく利用できるのが望ましい．伊達らはこれを透過性（transparency）とよんでいる．また，オンプレミス環境での実行結果とパブリッククラウド上での実行結果は一致しなければならない（伊達らはこれを同一性（equality）とよんでいる）．連携のための技術的要件を表 3 に示す．項番 1～3 についてはオンプレミス側とパブリッククラウド側のソフトウェアに関する実行環境を合わせ，かつ，両者のファイルシステムをネットワークを介して相互マウントすることで実現可能である．しかし，オンプレミス側ノードのコア数（または HW スレッド数）と同数コア以上の仮想マシンで実行するとき，両者のソケットあたりのコア数が異なる等の場合には使われない仮想マシンのコアがあったり，あるいは，実行性能が逆に落ちたりする可能性がある．また，実行ジョブのプログラム中に絶対パスで入出力ファイル名が指定されている場合には，ネットワークを介してオンプレミス側ファイルシステムにアクセスするから仮想マシン本来の性能が得られず，クラウド側の資源の有効利用という観点からも好ましくない．このため項番 7 を満たすのは難しい．

表 3　連携のための要件と可否

項番	技術的要件	Case 1	Case 2
1	オンプレミス環境で作成した実行モジュールと実行用シェルスクリプトがそのままパブリッククラウドで利用できる	○	○
2	入力ファイルは自動的にパブリッククラウドに転送される	○	○
3	オンプレミスのファイルシステム上で出力を取得できる	○	○
4	オンプレミス環境とパブリッククラウドでの実行結果が一致する	○	○
5	オンプレミス環境と同等以上のセキュリティが保証される	○	○
6	課金情報が取得できる	○	○
7	実行時間が同程度以下である	×	○

Case 2：利用者が積極的に利用する場合

ターンアラウンドタイムを短くしたいと考える利用者ならばパブリッククラウドを積極的に利用する可能性がある．例えば，図 14 のような構成でパブリッククラウドとの連携が可能であろう．表 3 の項番 1～3 については Case 1 と同様に両者のソフトウェアに関する実行環境を合わせ，かつ，ファイルシステムを相互マウントすることで実現可能である．ただし，項番 7 を満たすにはパブリッククラウドの利用環境を意識して仮想マシン本来の性能を引き出すような使い方が必要である．

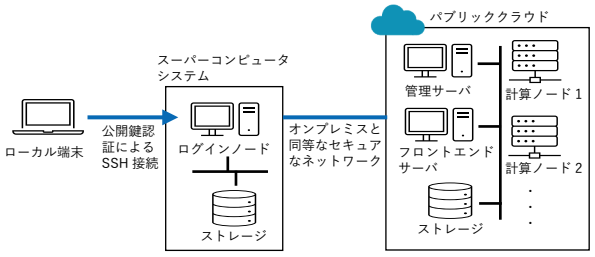

図 14　パブリッククラウドとの連携

表 4　仮想マシンの利用料金

VM サイズ	CPU	ソケット数	物理コア数	HW スレッド数	NVIDIA GPU	Inter-connect	利用料金（円/時間）			
							従量課金		スポット	
							East US	East Japan	East US	East Japan
HB120-16rs_v3	AMD EPYC 7V73X (Milan-X)	2	16	16	–	IB HDR 200Gbps	522.0	756.9	52.2	75.7
HB120rs_v3	AMD EPYC 7V73X (Milan-X)	2	120	120	–	IB HDR 200Gbps	522.0	756.9	52.2	75.7
HC44rs	Intel Xeon Platinum 8168 (Sky Lake)	2	44	44	–	IB EDR 100Gbps	459.4	666.1	45.9	66.6
NC24ads_A100_v4	AMD EPYC 7V12 (Milan)	1	24	24	A100 x 1	–	532.6	772.3	168.1	300.7
NC96ads_A100_v4	AMD EPYC 7V12 (Milan)	2	96	96	A100 x 4	–	2130.3	3088.9	672.3	1202.8
NC6s_v3	Intel Xeon E5-2690 v4 (Broadwell)	1	6	6	V100 x 1	–	443.7	608.1	206.7	213.2
NC24s_v3	Intel Xeon E5-2690 v4 (Broadwell)	2	24	24	V100 x 4	–	1774.8	2432.5	826.7	852.6
D32s_v5	Intel Xeon Platinum 8370C (Ice Lake)	1	16	32	–	–	222.7	287.7	36.5	29.1
D96s_v5	Intel Xeon Platinum 8370C (Ice Lake)	2	48	96	–	–	668.2	863.0	109.6	87.2
不老 Type I	Fujitsu A64fx	4	48	48	–	Tofu D	31.0 円/ノード時間			
不老 Type II	Intel Xeon Gold 6230 (Cascade Lake)	2	40	40	V100 x 4	IB EDR 200Gbps	38.8 円/ノード時間			
クラウド	Intel Xeon Gold 6230 (Cascade Lake)	4	80	80	–	IB EDR 100Gbps	44.3 円/ノード時間			

表 5　Azure NetApp Files の料金（円/4TiB・月）

Azure NetApp Files	単一暗号化		二重暗号化	
	East US	East Japan	East US	East Japan
Standard	87,579	104,922	104,055	150,879
Premium	174,725	209,410	208,110	301,759
Ultra	233,256	279,647	281,815	407,548

次に利用料金について考える．VM サイズごと仮想マシンの利用料金を表 4 に示す．表にはリージョン米国東部および東日本の従量課金とスポット課金を示した．スポットは，クラウド資源に余裕があるに場合のみ利用できる．キューイングされてもあとから従量課金ジョブ等の優先ジョブが入ってくると追い越される．さらに，スポットジョブが実行中でも優先ジョブの資源要求があると強制終了する．このときジョブスケジューラの機能によりに自動的に再キューイングされ計算資源が確保されると再実行される．ウェブサイト[14]より料金データをドル建てで取得し，1 ドル 145 円として換算した．表 4 には「不老」サブシステム Type I，Type II およびクラウドシステムのノード時間あたりの利用負担金も示されている．従量課金は高すぎて手が出ないが，スポット課金ならば可能性がありそうである（GPU 利用を除く）．また，一定期間の利用を確約し契約することで、仮想マシンの料金を従量課金よりも安く利用できるプランについても検討の余地がある．なお，利用料金にはパブリッククラウドではハードウェア，施設費，電気代，運用経費等が含まれるのに対して，「不老」には電気代と運用経費の一部が含まれる．

ストレージとして Azure NetApp Files を 1 か月間 4TiB 使用したときの利用料金[15]を表 5 に示す．NetApp Files は標準で FIPS 140-2 認定ドライブによるハードウェアベースの暗号化が提供され，さらにソフトウェアベースの暗号化を加えた二重暗号化が提供される[16]．なお， NetApp Files の二重暗号化は 2023 年 6 月にリリースされた機能で，本研究の性能測定では単一暗号化機能を使用した．「不老」のストレージを 1 か月間 4TB 使用したときの利用負担金は 1,385 円である．

Azure クラウドからデータを転送するときの料金[17]は，リージョンが米国東部で 1 か月あたり 10TB 転送するとき 116,000 円，リージョンが東日本の場合には 159,500 円である．

5 まとめ

本研究では Azure CycleCloud の利用環境と仮想ノード上でのベンチマークプログラム測定結果について報告した．そして，スーパーコンピュータシステムとパブリッククラウドの連携について考察した．本研究で得られたクラウドの利用経験を今後のセンター運営に活かしていきたい．

謝辞　本研究を実施するにあたってご協力をいただいた日本マイクロソフト社の方々に謹んで感謝の意を表します．

参考文献

[1] “名古屋大学情報基盤センタースーパーコンピュータシステム”. https://icts.nagoya-u.ac.jp/ja/sc/, (2023/08/22 参照).

[2] “クラウド バーストとは”. https://azure.microsoft.com/ja-jp/resources/cloud-computing-dictionary/what-is-cloud-bursting, (2023/08/22 参照).

[3] “Azure CycleCloud のドキュメント”. https://learn.microsoft.com/ja-jp/azure/cyclecloud/?view=cyclecloud-8, (2023/08/23 参照).

[4] “STREAM: Sustainable Memory Bandwidth in High Performance Computers”.

https://www.cs.virginia.edu/stream/, (2023/08/24 参照).
[5] “Performance & Scalability of HBv3 VMs with Milan-X CPUs”. https://techcommunity.microsoft.com/t5/azure-compute-blog/performance-amp-scalability-of-hbv3-vms-with-milan-x-cpus/ba-p/2939814, (2023/08/24 参照).
[6] “UL HPC MPI Tutorial: Building and Running OSU Micro-Benchmarks”. https://ulhpc-tutorials.readthedocs.io/en/latest/parallel/mpi/OSU_MicroBenchmarks/, (2023/08/24 参照).
[7] “HPC Performance and Scalability Results with Azure HBv3 VMs”. https://techcommunity.microsoft.com/t5/azure-compute-blog/hpc-performance-and-scalability-results-with-azure-hbv3-vms/ba-p/2206471, (2023/08/24 参照).
[8] M. Kawai et al., Hierarchical Parallelization of Multi-coloring Algorithms for Block IC Preconditioners, 2017 IEEE Proceedings of the 19th International Conference on High Performance Computing and Communications (HPCC 2017) held in Bangkok, 138-145, 2017.
[9] 大阪大学サイバーメディアセンター大型計算機システム OCTOPUS”. http://www.hpc.cmc.osaka-u.ac.jp/octopus/, (2023/08/25 参照).
[10] 伊達進他，OCTOPUS のクラウドバースティング拡張．大学 ICT 推進協議会 2019 年度年次大会, 2019.
[11] S. Date, et al., First Experience and Practice of Cloud Bursting Extension to OCTOPUS. Proceedings of the 10th International Conference on Cloud Computing and Services Science (CLOSER 2020), pp. 448-455, 2020
[12] A. Endo et al., Consideration of a Supercomputing System with Cloud Bursting Functionality from an Operational Perspective. 2022 IEEE International Conference on Cloud Computing Technology and Science (CloudCom), Bangkok, Thailand, pp. 154-161, 2022.
[13] “大阪大学サイバーメディアセンター大型計算機システム SQUID”. http://www.hpc.cmc.osaka-u.ac.jp/squid/, (2023/08/25 参照).
[14] “Linux Virtual Machines の料金”. https://azure.microsoft.com/ja-jp/pricing/details/virtual-machines/linux/#pricing, (2023/08/25 参照).
[15] “Azure NetApp Files の価格”. https://azure.microsoft.com/ja-jp/pricing/details/netapp/, (2023/10/02 参照).
[16] “Azure NetApp Files のドキュメント”. https://learn.microsoft.com/ja-jp/azure/azure-netapp-files/, (2023/10/02 参照).
[17] “帯域幅の価格”. https://azure.microsoft.com/ja-jp/pricing/details/bandwidth/, (2023/10/02 参照).

付録

付表 A　STREAM Triad ベンチマークテストでのコンパイラ等の組み合わせとその表記形式

表記形式	コンパイラ	コンパイルオプション	スレッドアフィニティに関係する環境変数
gcc9.2.0+threads+close	gcc 9.2.0	-march=native -O3 -fopenmp	OMP_PLACES=threads, OMP_PROC_BIND=close
gcc9.2.0+threads+spread	gcc 9.2.0	-march=native -O3 -fopenmp	OMP_PLACES=threads, OMP_PROC_BIND=spread
gcc9.2.0+cores+close	gcc 9.2.0	-march=native -O3 -fopenmp	OMP_PLACES=cores, OMP_PROC_BIND=close
gcc9.2.0+cores+spread	gcc 9.2.0	-march=native -O3 -fopenmp	OMP_PLACES=cores, OMP_PROC_BIND=spread
gcc9.2.0+sockets+close	gcc 9.2.0	-march=native -O3 -fopenmp	OMP_PLACES=sockets, OMP_PROC_BIND=close
gcc9.2.0+sockets+spread	gcc 9.2.0	-march=native -O3 -fopenmp	OMP_PLACES=sockets, OMP_PROC_BIND=spread
icc2023.0.0+thread+compact	icc 2023.0.0	-xHost -qopenmp -parallel	KMP_AFFINITY=granularity=thread,compact
icc2023.0.0+thread+scatter	icc 2023.0.0	-xHost -qopenmp -parallel	KMP_AFFINITY=granularity=thread,scatter
icc2023.0.0+core+compact	icc 2023.0.0	-xHost -qopenmp -parallel	KMP_AFFINITY=granularity=core,compact
icc2023.0.0+core+scatter	icc 2023.0.0	-xHost -qopenmp -parallel	KMP_AFFINITY=granularity=core,scatter
icc2023.0.0-zmm_high+core+compact	icc 2023.0.0	-xHost -qopenmp -parallel -qopt-zmm-usage=high	KMP_AFFINITY=granularity=core,compact
icc2023.0.0-zmm_high+core+scatter	icc 2023.0.0	-xHost -qopenmp -parallel -qopt-zmm-usage=high	KMP_AFFINITY=granularity=core,scatter
nvc23.1+threads+close	nvc 23.1	-fast -mp -Mconcur	OMP_PLACES=threads, OMP_PROC_BIND=close
nvc23.1+threads+spread	nvc 23.1	-fast -mp -Mconcur	OMP_PLACES=threads, OMP_PROC_BIND=spread
nvc23.1+cores+close	nvc 23.1	-fast -mp -Mconcur	OMP_PLACES=cores, OMP_PROC_BIND=close
nvc23.1+cores+spread	nvc 23.1	-fast -mp -Mconcur	OMP_PLACES=cores, OMP_PROC_BIND=spread