本論文は、2024 年 11 月公開済みの論文、電子情報通信学会ネットワークシステム研究会技術研究報告，“特定計算処理の量子コンピュータ処理オフロードの検討," の採録版のコピーである。

公開済み論文の書誌情報は、下記の通りである。
山登庸次, "特定計算処理の量子コンピュータ処理オフロードの検討," 電子情報通信学会ネットワークシステム研究会技術研究報告, NS2024-130, Nov. 2024. Copyright(C)2024 IEICE
https://ken.ieice.org/ken/paper/202411140cfw/

本論文が最初に公開された電子情報通信学会は、技術研究報告発行後のプレプリントサーバ登録を認めているため、Jxiv において公開することについて許可されている。

社団法人　電子情報通信学会
THE INSTITUTE OF ELECTRONICS,
INFORMATION AND COMMUNICATION ENGINEERS

信学技報
TECHNICAL REPORT OF IEICE.

特定計算処理の量子コンピュータ処理オフロードの検討

山登庸次†

† NTT，東京都武蔵野市緑町 3-9-11
E-mail: †yoji.yamato@ntt.com

あらまし　私達は，プログラマーが通常 CPU 向けに記述したコードを，配置される環境に応じて，自動で変換等をして，高性能に運用可能とする環境適応ソフトウェアを提案してきた．本稿は，新しいハードウェアへの自動オフロードの中で特に量子コンピュータを対象とする．オフロードしたい既存のアプリケーションを分析して，量子コンピュータに向いた計算として固有値解析問題等の特定の計算処理を発見し，その部分を量子コンピュータを利用する処理に置換することで，自動オフロードを実現する．提案方式で量子コンピュータに自動オフロードして実行できることを，Microsoft Azure Quantum で実際の処理時間を計測して確認する．

キーワード　環境適応ソフトウェア, 自動オフロード，量子コンピュータ，Qiskit，類似性検知

Study of quantum computer processing offload of specific calculation processing

Yoji YAMATO†

† NTT Corporation, 3-9-11, Midori-cho, Musashino-shi, Tokyo
E-mail: †yoji.yamato@ntt.com

Abstract　We have proposed an environment-adaptive software that automatically converts code written by programmers for regular CPUs according to the environment in which it is deployed, enabling high-performance operation. This paper focuses on quantum computers in particular as a part of automatic offloading to new hardware. We achieve automatic offloading by analyzing existing applications to be offloaded, discovering specific computational processes such as the eigenvalue solvers that are suitable for quantum computers, and replacing those parts with processes that use quantum computers. We confirm that the proposed method can be automatically offloaded to a quantum computer and executed by measuring the actual processing time on Microsoft Azure Quantum.

Key words　Environment Adaptive Software, Automatic Offloading, Quantum Computer, Qiskit, Similarity Detection

1. はじめに

近年，CPU の高速化を期待したムーアの法則の減速が予測されている．そのような状況から，CPU だけでなく，GPU（Graphics Processing Unit）や FPGA（Field Programmable Gate Array）等のハードウェアのシステムでの利用が増えている．例えば，Amazon 社は，GPU，FPGA サーバをクラウド（例えば, [1]- [6]）インスタンスとして提供しているし [7]. Microsoft 社は FPGA を使って Bing 検索効率を高めるといった取り組みをしており [8]，また，GPU や FPGA だけでなく，IoT 端末も増えているし（例えば, [9]- [16]），新しいハードウェアとして，量子コンピュータや DPU（Data Processing Unit）等も登場してきている．

しかし，通常コア数 CPU 以外のハードウェアを使ってシステムを高速化するためには，そのハードウェアの性質を意識したプログラム作成や設定が大前提であり，OpenMP（Open Multi-Processing）[17], CUDA（Compute Unified Device Architecture）[18]，OpenCL（Open Computing Language）[19] といった C 言語拡張等のプログラミングスキルが必要になってくるため，Python や Javascript といったスクリプト言語での開発が主流の大半のプログラマーにとっては，壁が高い．

通常コア数 CPU 以外の GPU や FPGA や，量子コンピュータ，DPU 等のヘテロジニアスなハードウェアを利用するシステムは，電力削減等の期待もあり，今後増えていくことが予想されるが，それらの実現にはスキルの壁が高い．そこで，スキルの壁を取り払い，通常コア数 CPU 以外のハードウェアを十

二分に利用できるようにするため，プログラマーが通常プログラムと同じ処理ロジックを記述したソフトウェアを，配置先の環境（GPU，FPGA，量子コンピュータ等）にあわせて，変換，設定し，環境に適応させるような，プラットフォームが重要になってくる．

そこで，私達は，記述された既存のソフトウェアコードを，配置先環境の GPU や FPGA 等を活用できるように，GPU や FPGA 向けの変換，リソース設定等を自動で行い，アプリケーションを高速に動作させる，環境適応ソフトウェアを提案した．合わせて，環境適応ソフトウェアの要素として，既存ソフトウェアのコードのループ文を，GPU，FPGA に自動オフロードする方式や一度オフロードした処理を運用中の状況変化に応じて再構成する方式を提案評価している [20]- [27].

しかし，私達の検証はこれまで，GPU や FPGA 等の既存のアクセラレータでの変換や設定，再構成が主流であり，量子コンピュータや DPU 等新しく出てきたハードウェアへの自動オフロードは検討されていなかった．例えば，量子コンピュータは現在，D-wave，Amazon，Microsoft，IBM 等数多くの会社が提供しており，NP（Non-deterministic Polynomial）問題への適用がされるようになっている．量子コンピュータは Amazon や Microsoft 等クラウドで時間貸しをしている会社もあり，膨大な初期費用が無くても利用できる．DPU は特定の Data 処理をアクセラレートする新たなハードウェアで，まだ少ないが利用も始まっている．

本稿は，新しいハードウェアへの自動オフロードの中で特に量子コンピュータを対象とする．特定処理高速化する DPU 等の新しいハードウェアにも適用可能な汎用的方式を目指す．まず，オフロードしたい既存のアプリケーションを分析して，量子コンピュータに向いた計算として固有値解析問題等の特定の計算処理を発見し，その部分を量子コンピュータを利用する処理に置換することで，自動オフロードを実現する．固有値解析問題の量子コンピュータでの解き方等，量子コンピュータアルゴリズムは本稿のターゲット外であり，本稿は固有値解析問題等の量子コンピュータに向いた問題を見つけ置換することがターゲットである．提案方式で量子コンピュータに自動オフロードして実行できることを，Microsoft Azure Quantum [28] で実際の処理時間を計測して確認する．

2. 既存技術

2.1 市中技術

GPU の計算力を画像処理以外にも使う GPGPU（General Purpose GPU）（例えば [29]）の環境として NVIDIA は CUDA を提供している．CUDA は GPU だけ対象としているが，GPU，FPGA，マルチコア CPU 等のヘテロジニアスハードウェアを同様に扱う仕様としては OpenCL があり，その開発環境は各社出している．CUDA，OpenCL は，C 言語拡張プログラムを行う形で，難度は高い（GPU 等のカーネルと CPU のホストとの間のメモリデータの解放やコピーの記述を行う等）．

CUDA や OpenCL に比べて，容易にヘテロジニアスハードウェアを利用するため指示行（Directive）を用いて，並列処理等を行う部分を指定して，指示行コマンドに従ってコンパイラが，GPU，マルチコア CPU 等に合わせたバイナリファイルを作成する技術がある．仕様としては，OpenMP や OpenACC [30] 等，コンパイラとして gcc や PGI コンパイラ [31] 等がある．

CUDA，OpenCL，OpenMP，OpenACC 等の技術を用いることで，GPU，FPGA，マルチコア CPU へオフロードすることは出来るようになっている．しかし処理自体は行えても，高速化には課題がある．例えば，マルチコア CPU 向け自動並列化として，Intel コンパイラ [32] がある．これらは，並列化時に，ループ文中で並列処理可能な部分を並列化する．しかし，メモリ処理等の影響で単にループ文を並列化しても性能がでない．GPU や FPGA 等で高速化する際には，CUDA や OpenCL 専門家がチューニングを繰り返したり，OpenACC コンパイラ等用いて適切な並列処理部分を探索することがされている．このため，スキルが乏しいプログラマーが，ヘテロジニアスハードウェアで高速化することは難しいし，自動並列化技術等を使っても並列処理部分試行錯誤等の稼働が必要だった．試行錯誤自動化として，著者は進化計算法用いた自動オフロードを提案している．

量子コンピュータは，量子力学原理を計算に応用したコンピュータである．D-wave 等は量子アニーリング法を用いているが，より汎用性が高い量子ゲート法での量子コンピュータの開発が主流になっており，Amazon，Microsoft，Google，IBM 等多くの企業が取り組んでいる．AWS Braket や Azure Quantum は，それぞれ Amazon の AWS クラウドや Microsoft の Azure クラウドの一つで提供しており，クラウドとして時間利用可能であり，量子コンピュータ自体のハードウェア初期費用は不要である．しかし，量子コンピュータを利用して NP 問題の巡回セールスマン問題や固有値解析問題を解く場合の量子アルゴリズムは専門家が研究して，一つ一つ考案しているのが現実であり，素人が利用することは難しい．

DPU は汎用 CPU とネットワーク・インターフェイス・ハードウェアを緊密に統合したプログラム可能なコンピュータープロセッサで，NVIDIA 社の BlueField [33] が著名である．従来の NIC（Network Interface Card）の代わりに使用し，メイン CPU の複雑なネットワーク処理やその他のインフラストラクチャ処理のオフロード等に利用される．

2.2 環境適応ソフトウェアの概要

環境適応ソフトウェアとして，私達は図 1 の処理ステップを提案している．環境適応ソフトウェアは，事業者が提供する，環境適応機能が中核に位置し，検証環境，商用環境，設備リソース DB，テストケース DB，コードパターン DB のモジュール群が連携することで動作する．

Step1　ユーザ提供コード分析：
Step2　オフロード可能部分抽出：
Step3　適切なオフロード部分探索：
Step4　商用リソース量調整：
Step5　商用配置場所調整：
Step6　バイナリファイル商用配置と検証：
Step7　商用運用中再構成：

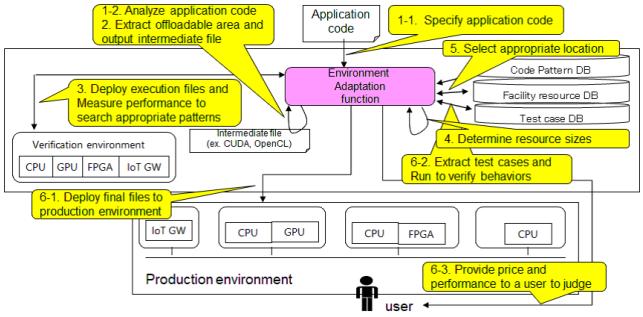


図 1　環境適応ソフトウェアの処理ステップ

最初にユーザに提供されたコードを分析し，Step1-6 で商用運用開始前に，検証環境での試験通じてコードの変換，リソース量の調整，配置場所の調整，検証を行う．Step7 は，商用運用開始後に，実用データを分析して，構成変更した方がコストパフォーマンス良い場合に再構成する．

2.3　本稿の課題

課題を整理する．ヘテロジニアスハードウェアでの高速化は専門家の手動が主流である．私は環境適応ソフトウェアを提案し，GPU や FPGA 自動オフロード方式も実現してきたが，量子コンピュータや DPU 等の新しいハードウェアへのオフロードは考慮されていない．そこで，本稿は，量子コンピュータを対象とする．DPU は検証しないが，特定処理をオフロードする DPU 等の新しいハードウェアにも適用可能な汎用的オフロードを行う．オフロードする内容は，ユーザが提供する既存の通常 CPU 向けの処理ロジックとする．専門家でないユーザが量子コンピュータを利用するのは市中に例が無く，難度，効果とも大きいため，まず量子コンピュータに取り組む．

3.　量子計算機への処理オフロード

私達は，環境適応ソフトウェアのコンセプトを具体化するために，これまでに，ループ文の GPU，FPGA 自動オフロードや，リソース量，配置の自動調整，運用中の再構成等を提案してきた．これらの要素技術検討も踏まえて，3.1 では，量子コンピュータ特性とそれを踏まえた基本検討について述べる．3.2 では，量子コンピュータ特性と基本検討を踏まえたオフロードに必要なオフロード部分発見について述べる．3.3 では，発見したオフロード部分の量子コンピュータ利用について述べる．

3.1　量子コンピュータ特性と基本検討

2 節で述べたように，現在多くの社が，量子ゲート法での量子コンピュータの開発に参画しており，提供される量子ビットも増加している．量子コンピュータは量子効果発現のため，希釈冷凍機での低温化がほぼ必須であり，自前で購入する場合は億円を超える額が必要である．初期費用無く量子コンピュータを利用するためには，Amazon Braket や Azure Quantum 等のクラウドとして提供されている量子コンピュータの利用が考えられる．

例えば，Azure Quantum のパフォーマンスコンピューティングであれば 1 時間辺り 90 USD 程度の価格である．そのため，これらクラウド型の量子コンピュータを利用するが，初期費用は無くても通常の CPU や GPU を備えた VM に比べると，時間当たり価格は高い．GPU や FPGA では，検証環境での繰返し試験を通じて段々速いパターンを見つける進化計算手法を使っていたが，量子コンピュータを検証のため長時間占有するのは費用上困難であるため，量子コンピュータにオフロードする際は，繰返し試験をせず効果がある計算だけ 1 回でオフロードすることが求められる．

量子コンピュータに効果がある計算をオフロードすることを考える．量子ゲート法での量子コンピュータの場合，巡回セールスマン問題，固有値解析問題，素因数分解等の組み合わせが膨大になる問題が通常 CPU に比べて格段に高速に解けると言われ，解き方は量子力学を理解した専門家が，各問題に適した量子アルゴリズムを設計している．

そこで，本稿では，ユーザがオフロードしたいコードの中で，量子コンピュータで効果がある計算部分を発見し，その計算部分を専門家が考案した量子アルゴリズムに従って，クラウド量子コンピュータを利用することで，自動オフロードを行う．

3.2　オフロードでの適した計算部分の発見

前サブ節の基本検討に従い，量子コンピュータに適した計算部分の発見について図 2 を参照して説明する．

A. 計算部分の発見ユーザが提供したコードの構文解析 Clang [34]，Lark [35] 等で行い，ソースコードから，クラス，構造体等の計算部分を検出する．中身は巡回セールスマン問題等実際に中身を見ないと問題の種類は分からないが，まず計算部分を検出する．

B. 量子コンピュータに適した計算部分の発見まず事前に，量子コンピュータに適した計算部分として，巡回セールスマン問題，固有値解析問題，素因数分解等を処理するコードを，コードパターン DB に保持しておく．更に，そのコードに該当する量子アルゴリズムで量子コンピュータを利用するための処理部分（ライブラリ相当）もコードパターン DB に保持しておく．

コードパターン DB に登録されている情報と照合して，量子コンピュータに適した計算部分を検出する．A で検出した，クラス，構造体等の計算部分に対して，オフロードできる量子アルゴリズムがあるかを，類似性検出ツールで検出する．類似性検出ツールとは，Deckard [36] や CloneDigger [37] 等，コピー後変更したコードのような類似コードの検出を可能とするツールである．類似性検出には，抽象構文木，語彙，行，プログラム依存グラフ，フィンガープリント等の類似性を用いるが，Deckard では抽象構文木を用いて，Java や C 言語のコードを解析する．勿論，量子コンピュータに適した計算部分が見つからない場合は多いと考えられるが，その場合は，その場合は検証環境での進化計算手法を用いた GPU 等での高速化と他のハードウェアでの高速化試行に入れば良い．

コードパターン DB に登録された，量子コンピュータに適した計算部分について，類似性が高いことをツールの閾値等で判定する．類似性検出でカバーできる範囲は 100%でないことは明らかなため，人工知能処理の深層学習認識等の適用も将来的な応用として考えられる．ただし，人工知能処理の場合も，機械にコードの意図を理解させることは困難なため，新規独立に

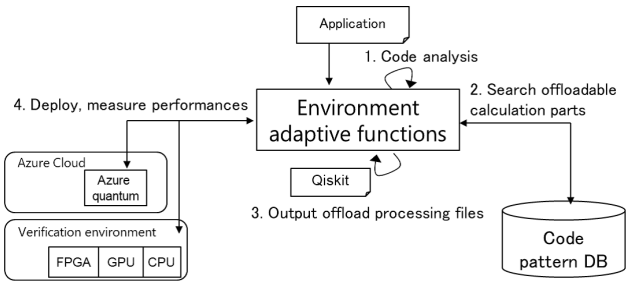


図 2　量子コンピュータへのオフロード処理

作成したようなクラス，構造体等については，量子コンピュータオフロードの対象外となると考える．

量子コンピュータに適した計算部分が発見された場合は，コードパターン DB に登録されている，そのコードに該当する量子アルゴリズムで量子コンピュータを利用するための処理部分（ライブラリ相当）を用いて，量子コンピュータにオフロードする．

3.3　量子コンピュータ処理

クラス，構造体等の計算部分に対して，量子コンピュータに適した計算かを検索しているため，適している場合はその処理部分（ライブラリ相当）をホスト側に置換して実装する．量子コンピュータは開発企業毎に異なるため，その処理も各社の提供ツールで実装するのが基本だが，各社で共通的に使えるフレームワークもある．Qiskit [38] は IBM がオープンソース化した量子コンピュータフレームワークで，Cirq は Google がオープンソース化した量子コンピュータフレームワークで，Q#は Microsoft が提案する量子コンピュータ用プログラム言語である．例えば，巡回セールスマン問題を含むユーザアプリケーションが指定された場合に，Deckard を使って，コードパターン DB から Qiskit で実装された量子アルゴリズムが発見され，それを用いて Azure Quantum に処理がされる．

ここで，特定の計算部分に対して量子コンピュータで処理する部分（ライブラリ相当）が見つかるため，インタフェース部分の生成等は必要になる．類似性で量子計算に適した計算か判断するので，Qiskit 等の実装部とホスト側プログラムの想定する引数や戻り値の数や型等の部分があっている保証はない．合っていない場合は，量子アルゴリズム実装部は変更が頻繁にできるものでないため，オフロード依頼するユーザに対して，元のコードの引数や戻り値の数や型について，量子コンピュータ処理部分に合わせて変更するか確認し，確認了承後にオフロード確認試験を試行する．型の違いについて，float と double 等キャストすればよいだけであれば，特にユーザ確認せずに確認試験に入ってもよい．また，引数や戻り値で，元のプログラムと量子コンピュータ処理部分で数が異なる場合に，例えば，CPU プログラムで引数 1, 2 が必須で 3 がオプションであり，量子コンピュータ処理部分で引数 1, 2 が必須の場合等，省略しても問題ないような場合は，ユーザに確認せず，オプション引数は自動で無しとして扱うなどしてもよい．

このように，ユーザがオフロード依頼するコードから，量子コンピュータに適した計算部分を類似性検知技術で検知し，オープンソースフレームワークで実装された量子コンピュータ処理部分に置換することで，量子コンピュータへの自動オフロードを行う．

4.　評　　価

ユーザが指定するアプリケーションから量子コンピュータへの自動処理オフロードができ，量子コンピュータ処理時間が適切であることを確認し方式の有効性を評価する．

4.1　評 価 条 件

4.1.1　評 価 対 象

評価対象は，固有値解析問題とする．高次の実数あるいは複素数の行列において，有限回の代数的操作によって固有値を厳密に表わす計算手順は存在しない．そのため，固有値問題の数値解析には必ず反復法を用いることになるのが従来である．量子コンピュータの量子状態重ね合わせの特性によって高速化が可能となる．一般の n 次代数方程式の同伴行列の固有値を求めることが，問題本質であり，各問題はパラメータの違いによる．利用する固有値解析問題は [39] とする．固有値解析問題は，[39] の通り量子コンピュータでの解法も量子アルゴリズムで Qiskit 実装がされている．

勿論，量子コンピュータは NP 問題のナップサック問題や素因数分解も量子アルゴリズムでの解法が研究されているが，汎用的問題で一般ユーザでの応用が広い固有値解析問題をオフロードの対象とする．

4.1.2　評 価 手 法

1 ユーザを想定し，ユーザは固有値解析問題を含むアプリケーションのオフロードを依頼する．事業者のプラットフォームは依頼を受けたらアプリケーションを分析し，オフロード計算部分を類似性検出ツールで検索し，発見出来たら量子コンピュータ利用部分に置換して，量子コンピュータにオフロードする．オフロード処理した際の処理時間を測定し，測定結果と利用価格がユーザに提示される．

実験の条件は以下で行う．

オフロード計算部分：固有値解析問題 [39]

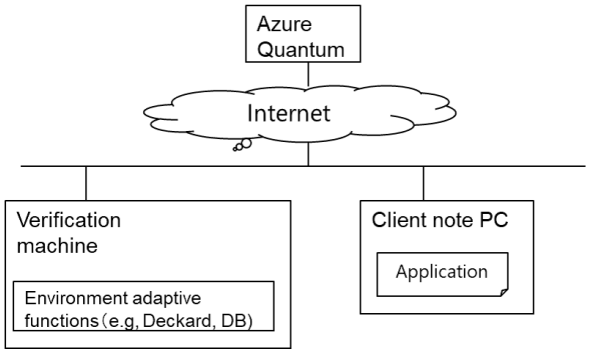
オフロード先：Azure Quantum

オフロード計算部分発見手法：アプリケーションのコードに含まれる計算部分を類似性検出ツール Deckard を用いて，コードパターン DB に含まれるコードと照合．

性能測定：[39] のサンプルパラメータそのままに，量子コンピュータでの処理時間を測定する．中程度サイズでは通常 CPU でも現実的時間で計算終了するが，今回は量子コンピュータオフロードを確認するため，大きいサイズでの計測はしない．

4.1.3　評 価 環 境

評価環境とスペックを図 3 に示す．ここで，Verfication machine を用いるノート PC のユーザは，オフロードしたいアプリケーションを指定し，Verification machine が Deckard を用いて同一ノード上の DB と照合し検索する．オフロードする際に，量子コンピュータ用の Qiskit 実装を取得し，インターネットを介して，Azure クラウド上の Azure Quantum に処理を依



Name	Hardware	CPU	RAM	OS
Verification machine	LEVEL-F039-LCRT2W-XYVI	AMD Ryzen Threadripper 2990WX	64GB	Ubuntu 16.04.6 LTS
Client note PC	HP ProBook 470 G3	Intel Core i5-6200U @2.3GHz	8GB	Windows 10 Pro
Quantum computer	Azure quantum			

図 3　性能測定環境

Application	Quantum computer	Processing time
Eigenvalue Solver	Azure Quantum	15.7 sec

図 4　固有値解析問題の量子コンピュータ処理時間

頼することでオフロードする．オフロード処理及び全体処理時間は Verfication machine で測定される．

4.2　結果と考察

図 4 は，提案方式で Azure Quantum にオフロードした際の処理時間を示している．依頼から期待通り Azure Quantum にオフロードしている．量子コンピュータの並列性から固有値解析問題のサイズが大きくなっても短時間で終わると期待されるが，今回でも全体で 15.7 秒で終わっている．オフロードするまでの時間自体は，ユーザリクエストから Deckard 検索，Qiskit 実装部の置換等，秒単位の処理で終了するため，特にユーザの待ち時間は発生しない．勿論，外部サービスとして，Azure を用いているため，Azure をサービス事業者でなくユーザ自身が契約する際は，審査等が必要なため，契約するまでに一定の時間と稼働が必要である．

以前の研究であるループ文の GPU，FPGA へのオフロードでは，複数のオフロードパターンを検証環境で性能測定し，高速なパターンにしていく手法で，例えば，for 文が 100 を超える大規模なアプリケーションである Darknet 等でも GPU に自動オフロードし 3 倍高速化をしてきた．しかし，GPU に対しては CUDA，FPGA に対しては HDL 等を用いて手動開発することで，10 倍以上高性能化している例は数多くあり，性能改善度が不十分の場合もあった．今回提案手法は，量子コンピュータに向いた計算部分を量子コンピュータに Qiskit でオフロードすることで，固有値解析問題等，サイズが大きくなると既存の CPU では解けないような問題を処理できている．

しかしながら，提案方式は，専門家が検討し，既に実装された Qiskit の処理部分の DB 登録を前提としているため，DB に登録されていない量子コンピュータに向かない計算部分については，オフロードすることはできない．そこで，以前研究のループ文の GPU や FPGA オフロード等の手法と組み合わせて利用することが実運用では必要になると考える．

コストパフォーマンスについて考察する．量子コンピュータをクラウドで提供している事業者はまだ少ないが，例えば Azure Quantum は 1 時間 90USD 程度で利用できる．そのため，1 か月フルで利用すると 65,000USD 程度かかり現実的でない．ユーザは，1 時間だけ使い解を計算すると言った使い方が現実的となる．一方で，サイズが小さければ CPU の VM でも現実的時間で解が出るため，量子コンピュータでなければ解けないサイズの問題を 1 時間だけ使う等のユースケースでコスト効果が出てくると言える．

サービス利用開始までの時間については，量子コンピュータオフロードの場合は秒単位で処理が終わると想定される．オフロードできる計算部分がない場合は，量子アルゴリズムを専門家が検討する際は数か月のオーダーで時間がかかるため，ループ文の GPU や FPGA オフロード手法での高速化を行い，数時間から 1 日の検証期間で高速化を行う．私達は，サービスを提供する際に，最初の一日は試用させ，その間に検証環境で高速化を試行し，2 日目から GPU，FPGA，量子コンピュータ等を用いた本番サービスを提供できるようにすることを考えているため，本提供開始まで 1 日かかるのは許容範囲と考える．

提案手法で量子コンピュータにオフロードするためには，多くのアプリケーションで共通的に使われる計算部分の Qiskit 実装等の DB への事前登録が必要となる．量子コンピュータ自体は，量子力学の並列性を活かした，巡回セールスマン問題，固有値解析問題，素因数分解等が適していると言われ，特定の計算領域に特化して登録されることを想定している．

また，パラメータ違いや少し変更したようなコードは抽象構文木の類似性検知で発見できるが，どういった範囲まで発見できるかをより細かく検証する．類似性検知に加えて，人工知能分野の深層学習処理等により，量子コンピュータにオフロードできる計算部分をより広く発見できることを検討する必要もある．

5. ま と め

本稿では，私達が提案している，環境適応ソフトウェアの新たな要素として，ユーザの提供するアプリケーションの処理ロジックを分析し，新たなハードウェアである量子コンピュータに適切な部分の処理を自動オフロードする方式を提案した．

量子コンピュータは，量子力学の重ね合わせの特性から，巡回セールスマン問題等の問題に適していると言われる．そこで，ユーザからオフロードしたいアプリケーションを指定されたら，量子コンピュータにオフロードするのが適切な計算処理が無いかを探索する．探索には，抽象構文木の類似度を用いて，具体的には Deckard を用いる．Deckard で照合される DB には，量子コンピュータに適切な計算処理のコードと，それに対応する量子コンピュータ処理用の Qiskit モジュールが登録され

ている．抽象構文木の類似度で量子コンピュータに適切な計算処理部分が見つかったら，それに対応する Qiskit モジュールに置換することで利用する．今回検証では，オフロードする計算処理を固有値解析問題とし，オフロード先を Azure Quantum とし，実際に Deckard でオフロード計算処理を発見し，Qiskit に置換して，Azure Quantum に処理をオフロードして，処理時間 15.7 秒で処理が終わり，方式有効性を確認した．

文　献

[1] O. Sefraoui, et al., "OpenStack: toward an open-source solution for cloud computing," International Journal of Computer Applications, Vol.55, No.3, 2012.

[2] Y. Yamato, "Automatic Verification for Plural Virtual Machines Patches," The 7th International Conference on Ubiquitous and Future Networks (ICUFN 2015), pp.837-838, July 2015.

[3] Y. Yamato, "Cloud Storage Application Area of HDD-SSD Hybrid Storage, Distributed Storage and HDD Storage," IEEJ Transactions on Electrical and Electronic Engineering, Vol.11, Issue.5, pp.674-675, DOI:10.1002/tee.22287, Sep. 2016.

[4] Y. Yamato, et al., "Development of Template Management Technology for Easy Deployment of Virtual Resources on OpenStack," Journal of Cloud Computing, Springer, Vol.3, No.1, DOI: 10.1186/s13677-014-0007-3, June 2014.

[5] Y. Yamato, "Use Case Study of HDD-SSD Hybrid Storage, Distributed Storage and HDD Storage on OpenStack," 19th International Database Engineering & Applications Symposium (IDEAS '15), pp.228-229, July 2015.

[6] Y. Yamato, et al., "Fast Restoration Method of Virtual Resources on OpenStack," IEEE Consumer Communications and Networking Conference (CCNC 2015), pp.607-608, Jan. 2015.

[7] AWS EC2 web site, https://aws.amazon.com/ec2/instance-types/

[8] A. Putnam, et al., "A reconfigurable fabric for accelerating large-scale datacenter services," Proceedings of the 41th Annual International Symposium on Computer Architecture (ISCA'14), pp.13-24, June 2014.

[9] M. Hermann, et al., "Design Principles for Industrie 4.0 Scenarios," Rechnische Universitat Dortmund. 2015.

[10] H. Noguchi, et al., "Autonomous Device Identification Architecture for Internet of Things," 2018 IEEE 4th World Forum on Internet of Things (WF-IoT 2018), pp.407-411, Feb. 2018.

[11] H. Noguchi, et al., "Distributed Search Architecture for Object Tracking in the Internet of Things," IEEE Access, DOI: 10.1109/ACCESS.2018.2875734, Oct. 2018.

[12] M. Takemoto, et al.,"Service-composition method and its implementation in service-provision architecture for ubiquitous computing environments," IPSJ Journal, Vol.46, No.2, pp.418-433, Feb. 2005.

[13] Y. Yamato, et al., "Proposal of Shoplifting Prevention Service Using Image Analysis and ERP Check," IEEJ Transactions on Electrical and Electronic Engineering, Vol.12, Issue.S1, pp.141-145, June 2017.

[14] Y. Yamato, et al., "Method of Service Template Generation on a Service Coordination Framework," 2nd International Symposium on Ubiquitous Computing Systems (UCS 2004), Nov. 2004.

[15] Y. Yamato, et al., "Ubiquitous Service Composition Technology for Ubiquitous Network Environments," IPSJ Journal, Vol.48, No.2, pp.562-577, Feb. 2007.

[16] P. C. Evans and M. Annunziata, "Industrial Internet: Pushing the Boundaries of Minds and Machines," Technical report of General Electric (GE), Nov. 2012.

[17] T. Sterling, et al., "High performance computing : modern systems and practices," Cambridge, MA : Morgan Kaufmann, ISBN 9780124202153, 2018.

[18] J. Sanders and E. Kandrot, "CUDA by example : an introduction to general-purpose GPU programming," Addison-Wesley, 2011.

[19] J. E. Stone, et al., "OpenCL: A parallel programming standard for heterogeneous computing systems," Computing in science & engineering, Vol.12, No.3, pp.66-73, 2010.

[20] Y. Yamato, "Automatic Offloading Method of Loop Statements of Software to FPGA," International Journal of Parallel, Emergent and Distributed Systems, Taylor and Francis, DOI: 10.1080/17445760.2021.1916020, Apr. 2021.

[21] Y. Yamato, "Study and Evaluation of Automatic GPU Offloading Method from Various Language Applications," International Journal of Parallel, Emergent and Distributed Systems, Taylor and Francis, DOI: 10.1080/17445760.2021.1971666, Sep. 2021.

[22] Y. Yamato, "Improvement Proposal of Automatic GPU Offloading Technology," The 8th International Conference on Information and Education Technology (ICIET 2020), pp.242-246, Mar. 2020.

[23] Y. Yamato, "Proposal and evaluation of GPU offloading parts reconfiguration during applications operations for environment adaptation," Journal of Network and Systems Management, Springer, DOI: 10.1007/s10922-023-09789-2, Nov. 2023.

[24] Y. Yamato, "Study and Evaluation of Automatic Offloading Method in Mixed Offloading Destination Environment," Cogent Engineering, Taylor & Francis, Vol.9, Issue 1, DOI: 10.1080/23311916.2022.2080624, June 2022.

[25] Y. Yamato, "Proposal of Automatic Offloading for Function Blocks of Applications," The 8th IIAE International Conference on Industrial Application Engineering 2020 (ICIAE 2020), pp.4-11, Mar. 2020.

[26] Y. Yamato, "Proposal of Automatic GPU Offloading Method from Various Language Applications," The 9th International Conference on Information and Education Technology (ICIET 2021), pp.400-404, Mar. 2021.

[27] Y. Yamato, "Study and Evaluation of Improved Automatic GPU Offloading Method," International Journal of Parallel, Emergent and Distributed Systems, Taylor and Francis, DOI: 10.1080/17445760.2021.1941010, June 2021.

[28] Azure quantum website, https://azure.microsoft.com/products/quantum

[29] J. Fung and M. Steve, "Computer vision signal processing on graphics processing units," 2004 IEEE International Conference on Acoustics, Speech, and Signal Processing, Vol. 5, pp.93-96, 2004.

[30] S. Wienke, et al., "OpenACC-first experiences with real-world applications," Euro-Par 2012 Parallel Processing, pp.859-870, 2012.

[31] M. Wolfe, "Implementing the PGI accelerator model," ACM the 3rd Workshop on General-Purpose Computation on Graphics Processing Units, pp.43-50, Mar. 2010.

[32] E. Su, et al., "Compiler support of the workqueuing execution model for Intel SMP architectures," In Fourth European Workshop on OpenMP, Sep. 2002.

[33] NVIDIA BlueField website, https://www.nvidia.com/networking/produc processing-unit/

[34] Clang website, http://llvm.org/

[35] Lark website, https://lark-parser.readthedocs.io/en/stable/

[36] Deckard website, http://github.com/skyhover/Deckard

[37] CloneDigger website, https://pypi.org/project/clonedigger/

[38] Qiskit website, https://www.ibm.com/quantum/qiskit

[39] Eigenvalue Solver website, https://learning.quantum.ibm.com/course/va algorithm-design/instances-and-extensions