量子回路シミュレータの性能分析と性能向上のための一検討

青木 望美 (指導教員：小口 正人)

1 はじめに

近年研究・開発が盛んに行われている量子コンピュータは，量子化学計算や組み合わせ最適化問題などの一部の問題を，従来型コンピュータと比べて非常に高速に解くことができるとされている．その理由に，量子コンピュータが扱う量子ビットがある．従来型コンピュータ上では情報の最小単位として 0 または 1 のビットを用いるが，量子コンピュータ上では 0 と 1 が重なり合った状態も保持可能な量子ビットを用いる．これにより同時に複数通りの計算方法を実行可能となり，従来型コンピュータと比較して非常に高速な計算を実現する．しかし，現在すでに存在する複数の量子コンピュータはどれもノイズの影響を受けてしまい，計算結果は正確なものを得ることが難しい．また，常に正確な挙動をする量子コンピュータの実現は数十年ほどかかるとも言われている．そのため，実際に量子コンピュータ上で使用することを想定した量子アプリ開発等を量子コンピュータ開発と並行して行うために，エラーなどの不具合が含まれない開発環境を用意することは重要である．

ここで，エラーなどが含まれない量子コンピュータ環境を従来型コンピュータ上で表現するツールに量子回路シミュレータというものがある．量子回路シミュレータを使用すると量子アプリなどの量子コンピュータ用ツールの開発を促進することができるため，量子回路シミュレータの性能向上手法に関する研究を行うことは有用である．

本稿ではいくつかある量子回路シミュレータの中でも，Qiskit Aer[1] という量子回路シミュレータについて，プログラム実行中のサーバシステム稼働情報を収集し，収集したデータを使用してボトルネックの要因を推測する．また，その結果から量子回路シミュレータの性能向上手法について検討する．

2 実験

2.1 実験概要

前章で記述した通り，分析対象の量子回路シミュレータは Qiskit Aer とする．実験の際は Qiskit Aer ライブラリを使用した Python プログラムを動作させ，複数の性能分析ツールを使用して実行中のサーバシステム稼働情報のデータを収集する．

今回用意した Python プログラムは量子体積モデル回路を用いている．量子体積は量子コンピュータのシステムを評価する目的で，どの程度複雑な処理が可能かを指標として提案 [2] された．量子体積モデル回路はこの量子体積を評価するためのものであり，ランダムな組み合わせの量子ビットのペアに 2 量子ゲートを作用させる．今回の実験では，量子ビット数 31 個，回路の深さが 10 の量子回路を使用する．

性能分析ツールは mpstat，vmstat，Linux Perf[3]，PCM Tools[4] を使用する．特に，Linux Perf は指定したイベントについてのデータを収集することができる perf stat と，全体実行時間の中で実行時間の消費率が高い関数を調べることができる perf record，perf report を使用し，PCM Tools は pcm-memory を使用してプログラム実行中のメモリバンド幅を計測する．なお，Python プログラムの実行開始は各性能分析ツールの実行開始から約 10 秒後とする．ただし，perf record のみ Python プログラムの実行コマンドを指定してデータ収集を行い，生成されるファイル容量が大きくなりすぎないよう量子ビットの個数を 27 個，回路の深さを 10 に変更する．

表 1 に実験環境，表 2 に性能分析ツールのバージョンを示す．mpstat は sysstat パッケージに含まれるツールのため，sysstat のバージョンを表記している．

表 1: 実験環境

実験用サーバ	Primergy RX2540 M1
OS	Rocky Linux 8.6
CPU	Intel Xeon プロセッサー E5-2697v3(2.60GHz)
メモリ	128GB

表 2: 性能分析ツールのバージョン

sysstat	11.7.3
vmstat	procps-ng 3.3.15
perf	4.18.0-425.3.1.el8.x86_64
pcm	202107-5.el8

2.2 結果

分析対象の Python プログラムの実行時間は約 261 秒であった．

はじめに mpstat の出力結果について記す．複数項目のうち，特にユーザ時間の割合を示す%usr について，すべての CPU コアでその値が常に 100 %近い値となった．よって，分析対象の Python プログラムは CPU コア単位での並列実行処理が行われていると考えられる．図 1 に全 CPU コアの%usr 値の平均のグラフを示す．

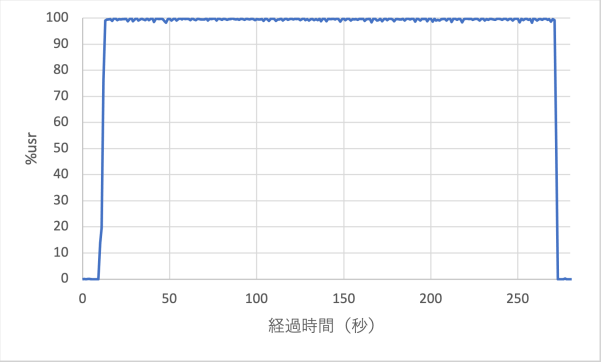

図 1: 全 CPU コアの%usr 平均値の変化

続いて vmstat の出力結果について記す．vmstat の出力には複数項目あり，そのうち空きメモリ量を出力する free の項目より，Python プログラムの実行中は約 33.7GB 相当のメモリが確保されていることがわかった．

次に perf stat での分析結果について記す．perf stat では収集するイベントデータを指定することが可能なため，データ収集は複数回に分けて行った．このとき複数回行った計測のいずれも，搭載されたすべての CPU コアに関してコアサイクルあたりの実行命令数（IPC）は常に 3 前後と，比較的高い値を保った．一例として，図 2 にコア ID=2 の CPU コアの IPC 値に関するグラフを示す．

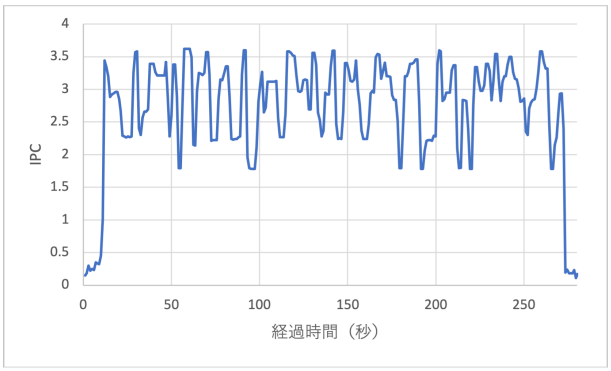

図 2: IPC の変化（コア ID=2）

さらに，perf stat を用いて L1 キャッシュミス率，L2 キャッシュミス率，L3 キャッシュミス率をそれぞれ計測したところ，Python プログラム実行中の CPU 全コアに関する平均値はそれぞれ約 1.7 %，約 45.2 %，約 65.4 %であった．

続いて perf record と perf report の出力結果について記述する．全体の実行時間の中で，apply_matrix_n という状態ベクトルを更新する関数が 96.93 %もの時間を消費していることがわかった．したがって，Qiskit Aer シミュレータの実行におけるホットスポット箇所はこの関数部分であると考えられる．

最後に pcm-memory での計測結果について記述する．Python プログラム実行中，メモリバンド幅はかなりの変動があるものの，基本的に 20 GB/s から 40 GB/s の範囲内で変化した．図 3 にメモリバンド幅の変化に関するグラフを示す．

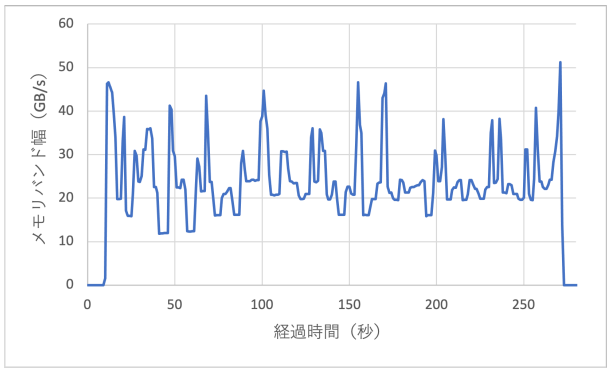

図 3: メモリバンド幅の変化

また，実験に使用したサーバのメモリバンド幅の理論値は 68 GB/s であるが，実効的なメモリバンド幅を引き出すことが可能な STREAM ベンチマーク [5] を実行し，pcm-memory を用いてメモリバンド幅を計測したところ，実行中の最大メモリバンド幅は約 57 GB/s となった．この値は Python プログラムのメモリバンド幅よりも大きい．今回分析対象としているシミュレータは State Vector 方式を採用しており，これは連続したアドレスに繰り返しアクセスする挙動を示すため，メモリバンド幅を改善することができる可能性が高い．

以上の結果から，Qiskit Aer シミュレータについてはメモリバンド幅をさらに引き出すことで性能の向上が期待できると考えられる．

3　まとめと今後の課題

本稿では量子回路シミュレータのひとつである Qiskit Aer シミュレータについて，そのライブラリを使用した Python プログラムを用意し，複数の性能分析ツールを使用してサーバシステムに関するデータの収集と分析を行った．その結果，今回分析したシミュレータの命令実行効率は良く，ホットスポット箇所は状態ベクトルの更新をする関数であり，さらに実効メモリバンド幅との比較からメモリバンド幅を引き出しきることができていない可能性があることがわかった．シミュレータの挙動を考慮するとメモリバンド幅を改善することができる可能性が高く，またこれを改善することでシミュレータの性能向上に繋げることができると考えられる．

今後は Qiskit Aer シミュレータのさらに詳細な分析を行うとともに，他の量子回路シミュレータについても同様の分析を行い，性能の比較や各シミュレータの性能向上のための具体的な手法を検討する．また，検討した性能改善手法を実際に検証する．

謝辞

本研究の一部はお茶の水女子大学と富士通株式会社との共同研究契約に基づくものである．

参考文献

[1] Qiskit aer simulator. https://qiskit.org/documentation/tutorials/simulators/1_aer_provider.html.

[2] A. W. Cross, L. S. Bishop, S. Sheldon, P. D. Nation, and J. M. Gambetta. Validating quantum computers using randomized model circuits. *Phys. Rev. A*, Vol. 100:p.32328(online), 2019. DOI: 10.1103/PhysRevA.100.032328.

[3] Perf wiki. https://perf.wiki.kernel.org/index.php/Main_Page.

[4] Intel® performance counter monitor - a better way to measure cpu utilization. https://www.intel.com/content/www/us/en/developer/articles/technical/performance-counter-monitor.html.

[5] Stream: Sustainable memory bandwidth in high performance computers. https://www.cs.virginia.edu/stream/.