TSUBAME 共同利用　令和7年度　学術利用　成果報告書

TSUBAME 4.0 における Quantum ESPRESSO の性能評価
Performance analysis of Quantum ESPRESSO in TSUBAME 4.0

太田 幸宏
Yukihiro Ota

高度情報科学技術研究機構 神戸センター 利用支援部
Department of HPC support, Kobe center, Research Organization for Information Science and Technology
https://www.hpci-office.jp/ristkobe/

TSUBAME 4.0 で Quantum ESPRESSO の性能評価を行う。特に、主要な計算エンジンである PWSCF に焦点を当てる。公開ベンチマークデータから SCF エネルギー1 点計算のデータを 3 種類選択し、GPU 数増加時のスケーラビリティ、k 点並列に関する並列設定が性能に与える効果を調べる。計測の結果、k 点並列の細かい制御を行うよりは、単純に GPU 数を増やすことでよい性能を得られることがわかった。さらに、PWSCF におけるピンメモリの利用について、コンパイルオプションにおけるグローバルな適用は注意が必要であることがわかった。

We examine the performance of a density-functional-theory-based quantum simulator for materials, Quantum ESPRESSO with use of GPUs in TSUBAME 4.0, focusing on the central computational engine, PWSCF. Choosing three kinds of benchmark data as SCF single-point energy calculations, we study the scalability as increase of the number of GPUs and effects of the runtime settings on k-point parallel executions. Our measurements indicate that good performance is attainable from a simple increase of the number of GPUs, rather than use of fine tuning of the k-point parallel executions. Moreover, we find that a global setting of compiler options on page-locked host memory can be a pitfall of performance.

Keywords: Quantum ESPRESSO, PWSCF, 密度汎関数法, 並列性能評価, ピンメモリ

背景と目的

Quantum ESPRESSO (QE) は物質の電子状態を計算するオープンソースソフトウェアのパッケージ群である[1]。物質・材料分野で幅広く使われており、拡張性も高く、他のコードとの連携もできる。QE では平面波基底での擬ポテンシャル密度汎関数法に従い、様々な物性値を量子論レベルで計算できる。

QE の MPI 並列性能はコマンドラインオプションを指定することで制御できる[2]。これはコミュニケータの種類に応じた設定や粒度を制御することに対応する。例えば、固有値ソルバーにおける行列の分割サイズ、高速フーリエ変換ライブラリに渡すデータサイズ、データ並列性が高い k 点ループの分割の制御が関連する。

GPU 搭載の並列計算機でプログラムを動作させるためには MPI プロセスと GPU とのバインディングの理解と制御が重要となる。QE では、HPC プログラムで典型的に採用されている方式、すなわち MPI ランクと GPU のデバイス ID を 1 対 1 で束縛させる方式 [3, 4] で実装されている。GPU 利用の観点から MPI 並列性能を制御するコマンドラインオプションを調整するための知見を得ることは、GPU 計算機で QE を利用する上で必要な技術情報になると考えられる。

本課題では、データサイズや利用する GPU 数に応じて、TSUBAME4.0 において QE を効率よく利用するための計算設定を調査する。また、QE の GPU 利用に対する実装を調査し、Fortran コードの GPU 化に対する知見を得ることを目指す。得られた結果や情報は QE を利用する様々な研究者にとって有意な情報になると期待される。

概要

表 1: 計測で使用したベンチマークデータ。QEF/benchmarks [7]から 3 種類を選択。

ベンチマーク名	ユニットセル内の原子数	カットオフ・エネルギー（波動関数）(Ry)	対角化手法	SCF 計算の最大反復回数	バンド数	FFT 3 次元グリッド数 (smooth)
AUSURF112	112	25.0	Davidson 法	2	800	1600000 (125,64,200)
GRIR443	443	30.0	Davidson 法	1	1793	2764800 (80,180,192)
PSIWAT	586	25.0	Davidson 法	5	1531	1600000 (64,125,200)

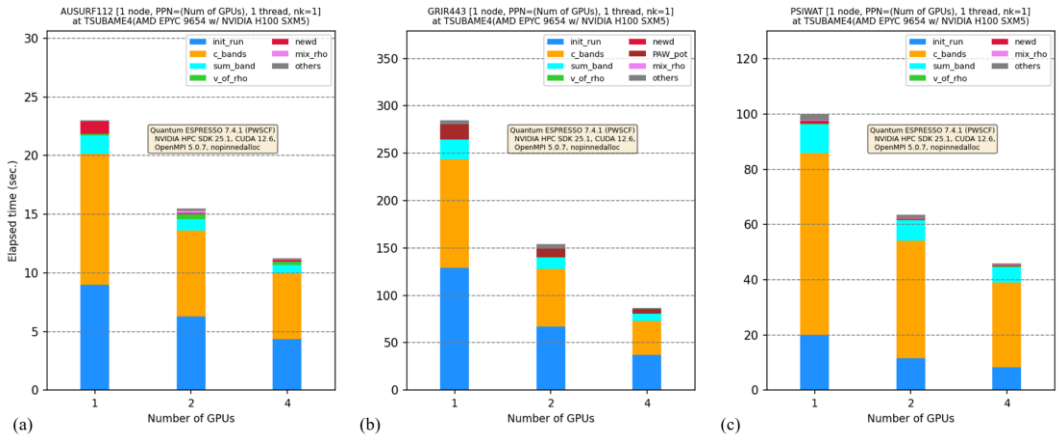

図 1: GPU 数（横軸）に対する PWSCF の経過時間（縦軸）。PPN は GPU 数に一致させ、1 OpenMP スレッドで実行。(a) AUSURF112、(b) GRIR443、 (c) PSIWAT。

TSUBAME 4.0 で QE の性能評価を行う。バージョンは 7.4.1 とする。QE のパッケージ群の中における主要な計算エンジンである PWSCF（平面波基底に基づく擬ポテンシャル密度汎関数法コード）に焦点を当てる。

QE の NVIDIA GPU 向け実装は OpenACC を主体としている。また、CUDA API を呼び出すために、CUDA Fortran で記述されたライブラリ deviceXlib [5] が使われている。なお、CP（Car-Parrinello 第一原理分子動力学コード）など一部のコードでは CUDA Fortran で実装されている。本利用課題で対象とする PWSCF は OpenACC とライブラリを組合させた実装となっている。OpenACC が実装の主体の場合、GPU のメモリ利用に関する最適化の余地は限られたものになる。本課題では、コンパイルオプションで制御できるレベルとしてピンメモリ（あるいは Page-locked host memory）[6] の効果を調べる。レファレンスとして、CPU のみの実行向けのバイナリで測定も行う。

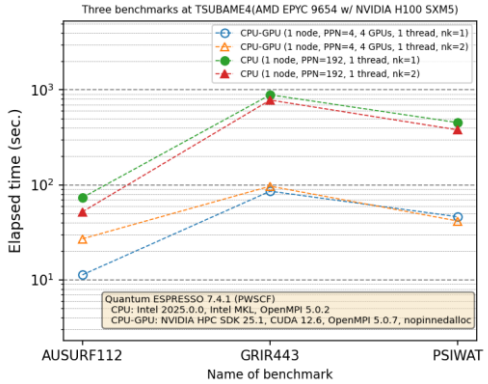

図 2: 3 種類のベンチマーク（横軸。表 1 を参照）に対する PWSCF の経過時間（縦軸）。

リスト 1: 区間 c_bands におけるホットスポットの典型例。PWSCF のコード[1]から抜粋。

```
! PW/src/vloc_psi_gpu.f90
ALLOCATE( psi(n,incr) )
ALLOCATE( psic(dffts_nnr*incr) )
!$acc data deviceptr(psi_d,hpsi_d) present(v, igk_k) create(psi,psic)
     ! ...
     !$acc parallel loop collapse(2)
     DO idx = 0, group_size-1
        DO j = 1, dffts_nnr
           psic(idx*dffts_nnr+j) = psic(idx*dffts_nnr+j) * v(j)
        ENDDO
     ENDDO
```

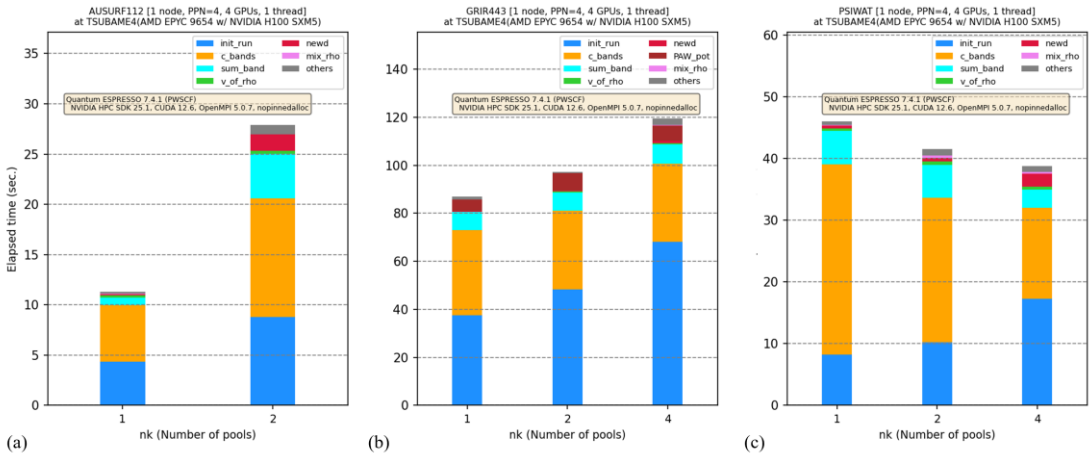

図 3: k 点並列自由度 nk の変更（横軸）に対する PWSCF の経過時間の内訳(縦軸)。TSUBAME4.0 を 1 ノード、PPN=4、4 GPUs、1 OpenMP スレッドで実行。(a) AUSURF112、(b) GRIR443、 (c) PSIWAT。

QE の性能評価では、PWSCF の典型例をデータの対象とする。データは公開ベンチマークデータ(QEF/benchmarks) [7] から選択して使用する。表 1 に使用したベンチマークデータをまとめる。

本利用課題では、MPI 並列設定に関するコマンドラインオプションとして、k 点ループの分割を制御する nk (Number of pools) を調べる。k 点並列に対応するループは高い並列性があるため、その特性を優先的に調べることに意義がある。k 点数は AUSURF112 で 2、他のデータで 4 のため、nk を AUSURF112 について最大で 2、他のデータについて最大で 4 まで変化させる。他のオプションとしては、ni (Number of images)、nt (Number of task groups) 、nd (Number of processors for linear algebra)、nb (Number of band groups) があるが、本利用課題ではプログラム側のデフォルト（自動設定）とした。なお、nt は FFT ライブラリに渡すデータサイズ（バンド数）と関わり、GPU 利用ではプログラム側でコンパイル時に固定される（16 バンド単位の処理がデフォルト)。そのため実行時の制御はできない。

性能分析は PWSCF の標準出力ログに表示される時間計測情報に基づく。こうして、コード内に設けられた計測区間の情報から経過時間の内訳を把握する。また、nsys profile によるサンプリング情報も利用する。計測時のコアバインディングは、OpenMPI のオプションで--bind-to core を指定する。なお、TSUBAME 4.0 では、QE のマルチノード実行は制限されているため[8]、本利用課題では 1 ノード内の調査に限定する。

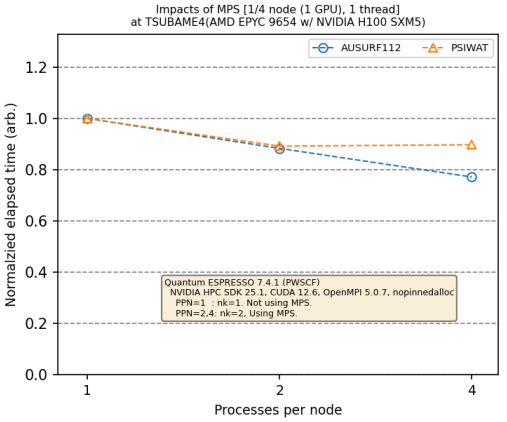

図 4: MPS の利用。1/4 ノードを使用。横軸は PPN。縦軸は PWSCF の経過時間。

結果および考察

表 1 に挙げたデータについて、密度汎関数法の理論レベルは一般化勾配近似で、Perdew-Burke-Ernzerhof 交換相関汎関数を用いる。擬ポテンシャルはベンチマークに同梱のものを利用する。全て、SCF によるエネルギー1 点計算に関するものである。ただし、設定された SCF 計算の反復回数(electron_maxstep)は少なく、収束していないことを確認している。また、計算部分の性能に着目するため、ファイル I/O 無しの設定（disk_io = 'none'）としている。表 1 には、計算設定に加え、データサイズを特徴づける量としてバンド数（Number of Kohn-Sham states)と FFT の 3 次元グリッド数（smooth grid）を記載している。

本利用課題で使用した QE の構築設定をまとめる。GPU 向けの構築環境は NVIDA HPC SDK 25.1 (cuBLAS,や cuFFT を使用)、CUDA 12.6、OpenMPI 5.0.7 を用いた。外部ライブラリとして、HDF5 1.14.3、libxc 5.1.2、(CPU 実行向けの) FFTW3 3.3.10 を用いた。OpenMPI とこれらの外部ライブラリは自身で構築した。最適化レベルは-fast とした。GPU のメモリ関連は、-gpu=mem:seperate:nopinnedalloc と -gpu=mem:separate:pinnedalloc の 2 種類を試行した。構築の際に CUDA-Aware MPI の利用が有効になるよう配慮した。CPU 向け QE の構築環境は、Intel oneAPI 25.0.0 (IntelMKL における BLAS/LAPACK、ScaLAKAPCK、FFT を使用)、OpenMPI 5.0.2 (システム提供版）を用いた。GPU 向けとの違いとして、ScaLAPACK を有効にしている点がある。最適化レベルは-O3 とした。GPU 版と同様に外部ライブラリ (HDF5、libxc) もリンクしている。

まず、3 種類のベンチマークに対し、nk=1 に固定し、1 ノードの範囲で GPU 数を増加させたときの計測結果を示す（図 1)。ノードあたりの MPI プロセス数（PPN）と GPU 数を一致させている。以後、断りがない限り、GPU 向けバイナリでは nopinnedalloc を設定している(すなわち、ピンメモリを QE 本体では利用しない)。横軸は GPU の数、縦軸は経過時間を示す。経過時間の内訳も区間ごとに示している。3 種類すべてのデータで、GPU 数に対してスケールすることがわかる。特に、1 GPU における高負荷計測区間 c_bands が、GPU 数の増大で削減される。区間 c_bands は SCF 計算の反復で呼び出され、GPU 化されたループに加え、固有値計算や FFT といった GPU 向けライブラリ・コールから構成されている。nsys profile のサンプリングにより、高負荷 GPU カーネルや API を把握することができる。リスト 1 に、区間 c_bands 中で高負荷 GPU カーネルを示す。OpenACC の parallel for で指定されたループが OpenACC カーネルとして最も高い負荷であり、その回転数はバンド数とプロセスあたりの FFT3 次元グリッド数で特徴づけられる。ただし、バンド数（リスト 1 の group_size）は 16 に固定されている。すなわち、外側にバンド数に関するループがあり、16 バンド単位で束ねて処理する。よって、データごとの特性という観点では、(MPI プロセスあたりの)FFT3 次元グリッド数(smooth grird) が重要となる（リスト 1 の dffts_nnr)。表 1 から、どのデータにおいても、プロセスあたり数十万以上の回転数が期待さ、H100 の利用に適したデータ規模であると推測される。図 1 の計測結果はそれを支持している。

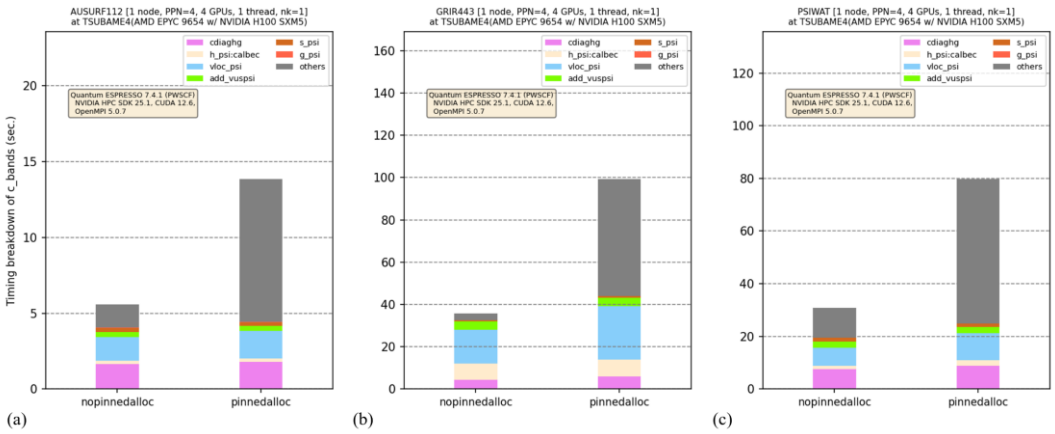

図 5: ピンメモリの効果。nopinnedalloc (pinndedalloc)でピンメモリをコンパイル時に無効（有効)。区間 c_bands に対する時間内訳を表示（縦軸)。1 ノード、PPN=4、4 GPUs、1 OpenMP スレッドで実行。nk=1 に固定。(a) AUSURF112、(b) GRIR443、 (c) PSIWAT。

コマンドラインオプション nk の変化に対する計測結果を調べる（図 2)。横軸はベンチマークの種類を示す。縦軸は経過時間を示す。白抜きマーカーが GPU 利用、それ以外は CPU のみを利用した場合の計測結果に対応する。コマンドラインオプション nk は 1 あるいは 2 とした。GPU 利用では 4 MPI プロセス/ノードと 4 GPUs、CPU のみの利用では 192 MPI プロセス/ノード（pure MPI）としている。AUSURF112 の nk=2 を除き、CPU のみに比べ GPU 利用で 8-10 倍程度の性能向上が確認される。AUSURF112 については、nk=2 では GPU での性能が劣化するため、他の設定に比べて性能向上は軽微である。

コマンドラインオプション nk の変更が性能に与える影響を理解するため、経過時間の内訳を調べる（図 3)。区間 init_run と区間 c_bands が経過時間の大きな部分を占めることがわかる。区間 init_run は nk>1 で増大しており、k 点並列に対する性能向上の阻害要因とみなせる。ただし、この区間は SCF 計算の反復ループの外側で呼び出されており、今回のような反復回数が少ない設定で見かけ上で顕著になった可能性はある。区間 c_bands については AUSURF112 では負荷増、GRIR443 ではほぼ一定、PSIWAT では負荷減という挙動が観察された。AUSURF112 では区間 c_bands の高速化が飽和することが k 点並列の効果を軽微なものにしていると言える。PSIWAT については、反復回数がより多い本格的な計算で nk の増大による性能向上の効果を得られる可能性がある。

以上の計測結果から、PWSCF の GPU 利用について得られた知見をまとめる。今回計測したデータの範囲では、k 点並列の細かい制御を行うよりは、単純に GPU 数を増やすことがよいと言える。並列設定はプログラム側のデフォルトとしておくのがよい。

次に、NVIDIA MPS の利用とコマンドラインオプション nk の設定の関係を調べる。ここまでの実行では、PPN と GPU 数を一致させてきた。MPI プロセスと GPU について 1 対1でバインディングを破ると性能が大きく劣化することを確認している（QE では、この条件を破ると oversubscription の警告がでる)。例えば、1 ノード 1GPU 利用（MIG による 1/4 キューに対応)で、AUSURF112 について、PPN=1 では経過時間は 22.92 秒に対し、PPN=4 では 75.19 秒であった。CPU 側リソースの有効利用という観点で、コード変更を必要としない対応である MPS を利用した。MPS は nvidia-cuda-mps-control -d をジョブスクリプト中で実行することで利用できる。TSUBAME 4.0 の 1/4 キューにおいて、MPS を有効にして計測を行った（図 4)。GPU 数は 1 とし、PPN=1, 2, 4 と変更する。PPN>1 に対し nk=2 とする。GRIR443 は GPU 側のメモリ不足に遭遇したため計測から除外している。図 1 に示した GPU 数を増大させたときの向上ではないが、AUSURF112 で k 点並列による性能向上の効果が出ている。図 3 の傾向と対照的である。このように、GPU

数を増やすことが難しい状況であれば、MPS による CPU 側リソースの利用と QE のコマンドラインオプションを組み合わせた高速化が可能なことがわかった。

最後に、コマンドラインオプション nk の調査から離れ、コンパイラオプションの影響を調べる。GPU のメモリ関連のオプションとして、ピンメモリの影響を調べる。図 5 に計測結果を示す。ここでは、PWSCF の経過時間の中でホットスポットである区間 c_bands に着目している。コンパイルオプションで pinnnedalloc を設定すると、標準出力ログに計測区間として現れない others の負荷が増大していることがわかる。nsys profile で計測したところ、cudaMemAllocHost が高負荷として測定された。コンパイルオプションで nonpinnnedalloc を設定することで、その負荷が大幅に減ることから、others の大部分はメモリ確保に起因すると推測される。nsys profile で計測したタイムラインから、リスト 1 の create 指示文で指定された配列（つまりホスト-デバイス間データ転送は行われないがページは確保）を利用する箇所で cudaMemAllocHost が呼び出される可能性があることもわかった。なお、リスト 1 は計測区間 vloc_psi に対応しており、others 以外での cudaMemAllocHost の影響が出た箇所である。実際、区間 vloc_psi の経過時間は nopinnedalloc の設定で短くなる。こうして、OpenACC のグローバルなコンパイルオプションとして pinnedalloc を設定することは注意が必要であることがわかった。ピンメモリは非同期処理などカーネルに応じたデータ特性を把握し CUDA による低レベル実装をすべきものである [6]。この特性を考えれば、結果そのものは自明ではあるが、コンパイルオプションの設定で大きく性能が変わる可能性を把握できたことは意義があると考えている。

まとめ、今後の課題

TSUBAME 4.0 で QE の主要計算エンジンである PWSCF について性能評価を行った。公開ベンチマークデータ 3 種類について、k 点並列の制御の観点から、GPU 計算機における MPI 並列性能を調べることができた。今回計測したデータの範囲では、k 点並列の細かい制御を行うよりは、並列設定はプログラム側のデフォルトとし、単純に GPU 数を増やすことがよい性能を得られることがわかった。ただし、GPU 数を増やせないときなどは MPS と k 点並列を併用することで、一定の高速化が得られることがわかった。さらに、PWSCF におけるピンメモリの利用について、コンパイルオプションにおけるグローバルな適用は注意が必要であることがわかった。

今後の課題として、今回の調査から外したコマンドラインオプションを取り上げることが考えられる。特に、バンド並列に関する自由度（nb）の制御は計算負荷が高い計算手法（例: ハイブリット汎関数の利用）で効果的な可能性がありある。CPU 計算機では性能向上に効果的だった経験があるため、GPU 計算機での効果を把握したい。また QE と外部コードを連携した場合の性能を評価したいと考えている。さらに、QE において GPU のメモリ関連の最適化に取り組む方法も調べたいと考えている。

参考文献

[1] Quantum ESPRESSO; http://www.quantum-espresso.org

[2] Quantum ESPRESSO User Guile: Parallelization levels; https://www.quantum-espresso.org/Doc/user_guide/

[3] HAIRDESC: GPU プログラミング教材「マルチ GPU プログラミング」; https://hairdesc.jp/gpumaterial/

[4] https://github.com/jirikraus/Multi_GPU_Programming_with_MPI_and_OpenACC

[5] deviceXlib; https://gitlab.com/max-centre/components/devicexlib

[6] CUDA Programming Guide, 2.4.3. Page-Locked Host Memory; https://docs.nvidia.com/cuda/cuda-programming-guide/

[7] QEF /benchmarks; https://github.com/QEF/benchmarks.git

[8] TSUBAME 4.0 利用の手引 7.1.7. QUANTUM

ESPRESSO;
https://www.t4.cii.isct.ac.jp/docs/handbook.ja/freesoft/#q-e