DFToy: A new proxy app for DFT applications

Arjen Tamerus
at748@cam.ac.uk
University of Cambridge
Cambridge, UK

Phil Hasnip
phil.hasnip@york.ac.uk
University of York
York, UK

ABSTRACT

Electronic structure codes based on density functional theory are a significant consumer of HPC resources, and play an important role in cutting-edge research on novel materials. As computing resources continue to increase these codes are used to investigate ever more complex materials. Unfortunately, DFT-based codes tend to be large, complex, and developed for past-generation hardware and can be hard to adapt to the current model of high-performance computing architectures.

This work introduces a new, low-complexity proxy-application for DFT codes that offers a low-access-barrier benchmarking platform, parallel scaling model, and experimental platform for the development of novel algorithms that can better exploit current hardware architectures.

ACM Reference Format:
Arjen Tamerus and Phil Hasnip. 2023. DFToy: A new proxy app for DFT applications. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 2 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

Materials modelling codes are some of the largest consumers of core hours on many TOP500-ranked supercomputers. Codes like VASP, CASTEP, Quantum Espresso and others frequently top the rankings when it comes to core hours consumed on the UK's national supercomputer ARCHER2[1], and many other systems show similar statistics.

These density functional theory based codes have applications in material discovery, the development of new battery materials, medical molecules, superconductor research and many more fields beside those. There is therefore no reason to expect the popularity and resource usage of these code to diminish anytime soon - if anything, the opposite is more likely: the number of citations of DFT and other atomistic codes is trending upwards[4][3].

Unfortunately, traditional DFT codes have some computational characteristics that limit their scalability to new Exascale or even large Petascale computing clusters. A further complicating factor is that these codes often have had decades of development on classical HPC infrastructures - making them robust, but also highly complex and hard to adapt to modern architectures.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
Conference'17, July 2017, Washington, DC, USA
© 2023 Association for Computing Machinery.
ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/XXXXXXX.XXXXXXX

2 DFT SCALING BOTTLENECKS

With the changes that the Exascale era has brought to HPC architectures, codes need to adapt to fully utilise the hardware. In CASTEP specifically, we have identified the following issues:

- High communication overhead: a significant portion of calculation time is spent transforming between real and reciprocal space - a global, all-to-all Fourier transform.
- Unfavourable memory access patterns: non-coalesced memory access and indirection.
- Reliance on double precision: increases memory pressure, and limits GPU compute to datacentre-level chips.
- High memory allocation/transfer/bandwidth pressure - limits effective use of accelerators and stresses interconnects.

3 DFTOY

In order to adapt to the modern HPC architectures of the Exascale era, the classic DFT codes will have to significantly adapt their code bases. Unfortunately, these codes tend to be large ($O(100000)$ to $O(1000000)$ LoC), dense, quite complicated - and often governed by a non-free license. This makes updating these codes, e.g. implementing effective GPU support, a non-trivial exercise especially without a solid understanding of the entire code bases and its underlying mathematical foundation.

We introduce DFToy[2], a new proxy-app that emulates the computational load of DFT codes. DFToy isolates the core computational load from DFT calculations in a simplified (currently a few thousand LoC) but representative code.

As currently implemented, DFToy implements a full DFT-alike (but simplified) calculation. It captures the most significant computational characteristics found in density functional theory codes - importantly, the expensive all-to-all communication pattern used by the Fast Fourier Transform that is used to convert between real and reciprocal space, and vice versa; and the $O(n^3)$ scaling of the dense matrix multiplications used in these codes.

4 THE ENGINE

DFToy's main computational load is finding the minimum energy of a fake 'atom' of "nonexistium". This is done by constructing the Hamiltonian matrix $\hat{H}$ from the semi-arbitrarily defined Kinetic energy and local and non-local potentials:

$$\hat{H} = \hat{T} + \hat{V}_{loc} + \hat{V}_{nl}$$

As DFToy is not concerned about real-world materials - we just want to model the scaling behaviour - we can significantly simplify the Hamiltonian's components.Each of the Hamiltonian's components is implemented as an $NxNxN$ matrix, representing a sampled force field with radius $r = 1$ and sampling resolution s, giving us $N = 2s + 1$ accounting for a zero point. We can change the

problem size by increasing the sampling resolution, i.e. the number of wavevectors w in our calculation ($w = (2s + 1)^3$).

We find our solution by finding the lowest M energy states - making this effectively an Eigensolver implemented with an iterative CG algorithm.

5 DFTOY VS CASTEP - PRELIMINARY BENCHMARKS

In order to investigate DFToy's scaling behaviour, we ran a benchmark on up to four nodes of the Icelake partition of the Cambridge supercomputer, CSD3. We compared a CASTEP 23 run of the Solid Benzene benchmark (figure 1) to a DFToy calculation using a wavevector resolution of 125, searching for the 8 lowest Eigenstates (figure 2).

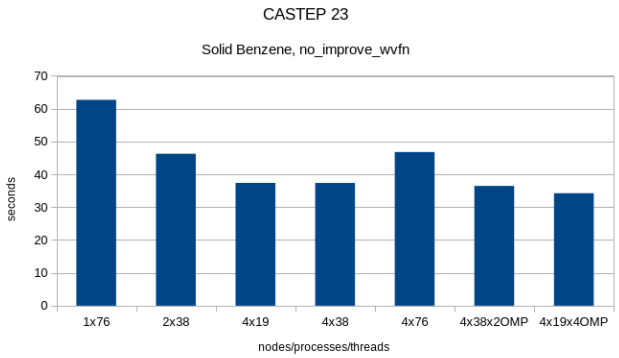

Figure 1: CASTEP: Solid benzene benchmark

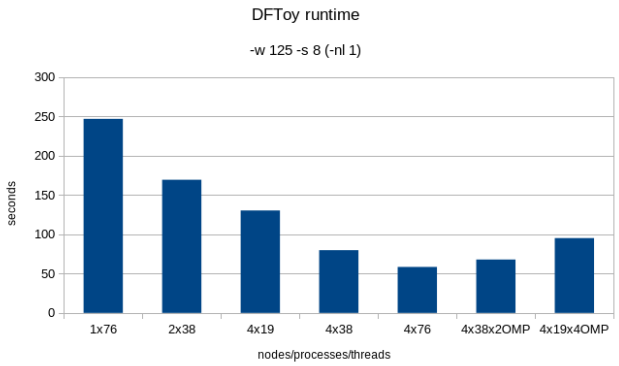

Figure 2: DFToy benchmark

While there is stil some work left to do on the Hybrid MPI/OpenMP implementation in DFToy - which shows in the reversed scaling behaviour of the OpenMP-enabled runs. The pure MPI benchmarks show a closer match to CASTEP's behaviour. Especially interesting is the increased performance when running on the same number of cores, but on higher node counts - i.e. reducing the per-node utilisation. This seems to indicate contention for memory bandwidth, which supports the claim that DFT codes are memory bandwidth hungry. Given the fact that CPU core counts seem to increase faster than CPU-to-RAM bandwidth, this is one of the areas that would be interesting to investigate with DFToy in the future.

6 FUTURE APPLICATIONS

With our first milestone - a functional DFT-alike proxy application with similar scaling behaviour to 'real' DFT codes - close to completion, we can look to future applications and features we are hoping to implement.

As DFToy is self-contained, free and open source, and requires no domain knowledge to run we consider it to be an excellent tool to use for performance and procurement benchmarking, and will working to make sure it's behaviour matches well enough to the 'big codes' to fulfil this promise. We also aim to develop a parallel behaviour model of the code, predicting the most efficient way to distribute it's calculation across parallel methods on any given hardware - and auto-tuning the application to run as efficiently as possible with minimal user input.

We will also use DFToy to investigate novel algorithms and the (more efficient) use of current and novel accelerators in an effort to avoid the current bottlenecks suffered by DFT calculations.

We hope to eventually bring any successful work in DFToy into CASTEP, to support efficient use of future HPC architectures and enable research into increasingly complex materials.

ACKNOWLEDGMENTS

This work was performed using resources provided by the Cambridge Service for Data Driven Discovery (CSD3) operated by the University of Cambridge Research Computing Service (www.csd3.cam.ac.uk), provided by Dell EMC and Intel using Tier-2 funding from the Engineering and Physical Sciences Research Council (capital grant EP/T022159/1), and DiRAC funding from the Science and Technology Facilities Council (www.dirac.ac.uk).

REFERENCES

[1] [n. d.]. ARCHER2 usage statistics - https://www.archer2.ac.uk/support-access/status.html#usage-statistics.
[2] [n. d.]. DFToy - https://github.com/ArjenTamerus/DFToy.
[3] [n. d.]. Leopold Talirz, Edoardo Aprà, Jonathan E. Moussa, & Samuel Poncé. (2023). ltalirz/atomistic-software: v2023.1.28 (v2023.1.28). Zenodo. https://doi.org/10.5281/zenodo.7578861.
[4] Leopold Talirz, Luca M. Ghiringhelli, and Berend Smit. 2021. Trends in atomistic simulation software usage [Article v1.0]. *Living Journal of Computational Molecular Science* 3, 1 (oct 2021). https://doi.org/10.33011/livecoms.3.1.1483

Received 12 August 2023