JCTC Journal of Chemical Theory and Computation

Article

pubs.acs.org/JCTC

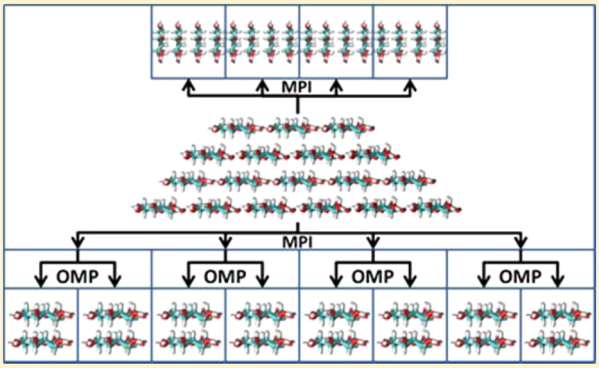
Hybrid MPI-OpenMP Parallelism in the ONETEP Linear-Scaling Electronic Structure Code: Application to the Delamination of Cellulose Nanofibrils

Karl A. Wilkinson,[†] Nicholas D. M. Hine,[‡] and Chris-Kriton Skylaris*,[†]

[†]School of Chemistry, University of Southampton, Southampton SO17 1BJ, United Kingdom

[‡]TCM Group, Cavendish Laboratory, University of Cambridge, JJ Thomson Avenue, Cambridge CB3 0HE, United Kingdom

ABSTRACT: We present a hybrid MPI-OpenMP implementation of Linear-Scaling Density Functional Theory within the ONETEP code. We illustrate its performance on a range of high performance computing (HPC) platforms comprising shared-memory nodes with fast interconnect. Our work has focused on applying OpenMP parallelism to the routines which dominate the computational load, attempting where possible to parallelize different loops from those already parallelized within MPI. This includes 3D FFT box operations, sparse matrix algebra operations, calculation of integrals, and Ewald summation. While the underlying numerical methods are unchanged, these developments represent significant changes to the algorithms used within ONETEP to distribute the workload across CPU cores. The new hybrid code exhibits much-improved strong scaling relative to the MPI-only code and permits calculations with a much higher ratio of cores to atoms. These developments result in a significantly shorter time to solution than was possible using MPI alone and facilitate the application of the ONETEP code to systems larger than previously feasible. We illustrate this with benchmark calculations from an amyloid fibril trimer containing 41,907 atoms. We use the code to study the mechanism of delamination of cellulose nanofibrils when undergoing sonification, a process which is controlled by a large number of interactions that collectively determine the structural properties of the fibrils. Many energy evaluations were needed for these simulations, and as these systems comprise up to 21,276 atoms this would not have been feasible without the developments described here.



1. INTRODUCTION

Methodological developments over the last two decades have made it possible to apply electronic structure methods to systems containing many thousands of atoms. A number of codes, such as ONETEP,[1] CONQUEST,[2] SIESTA,[3] OPENMX,[4] Ergo,[5] CP2K,[6] and BigDFT,[7] have been developed with calculations of this scale in mind, many of them based on linear-scaling formulations of density functional theory (LS-DFT).[8−11] However, such large numbers of atoms mean that the complexity of any technique requiring configurational sampling (eg geometry optimization, *ab initio* molecular dynamics, metadynamics, etc.) rapidly increases. The only way to utilize these methods at the scale of thousands of atoms is to take advantage of the massive parallelism made available by recent developments in high performance computing (HPC).

The aforementioned codes for large scale DFT calculations can run on these powerful new HPC platforms, but most development has focused on ensuring that so-called "weak scaling" is maintained, such that larger problems can exploit larger numbers of cores without loss of efficiency. Therefore, a single point energy evaluation of a given configuration can typically still take of the order of several hours. While acceptable for some techniques, this is much longer than the time-scale required to make methods such as *ab initio* molecular dynamics feasible. Unfortunately, it is not always straightforward to address this issue by increasing the number of cores used in a calculation: the parallel scaling of most codes for a fixed problem size will saturate above a relatively small number of cores.

Motivated by the need to improve upon this limit to the scaling, we present here new developments on the parallelization strategy of the ONETEP code that allow "strong scaling" (i.e., scaling with core count at fixed problem size) up to very large numbers of cores, even to many cores per atom, so that the wall time for a given calculation is significantly reduced - such improvements will also benefit the weak scaling of the code. These developments are implemented by the introduction of an open multiprocessing (OpenMP)[12] level of parallelism, below the existing message passing interface (MPI),[13] resulting in a hybrid MPI-OpenMP implementation.

A hybrid implementation maps well to modern supercomputer designs where the shared memory space on an

Received: April 15, 2014

dx.doi.org/10.1021/ct500686r | *J. Chem. Theory Comput.* XXXX, XXX, XXX−XXX

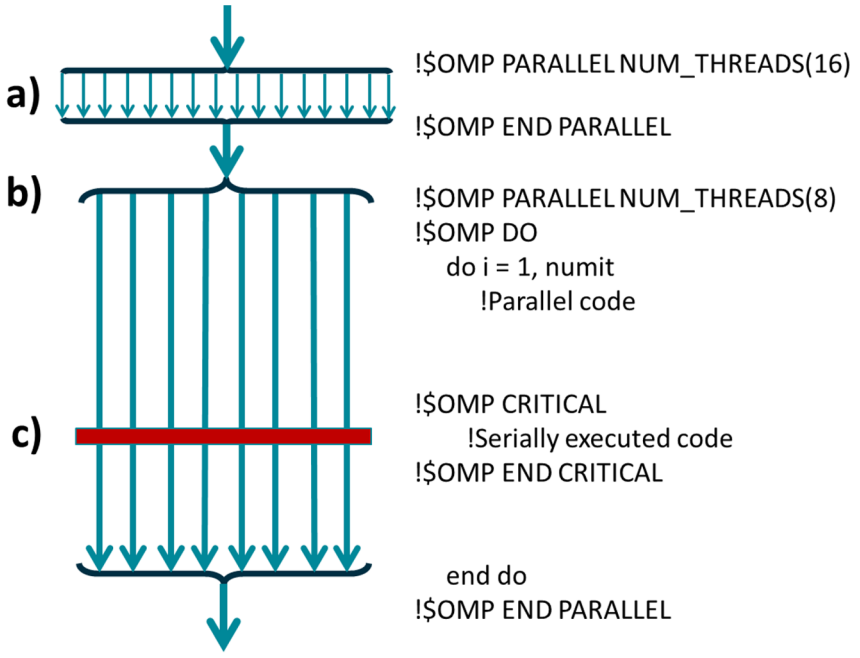


Figure 1. Example of OpenMP usage within an MPI process in ONETEP. (a) and (b) illustrate how the number of threads created in different OpenMP regions can vary, showing that in this example 16 threads have been chosen to be used for region (a) while 8 threads are used for region (b). (c) illustrates that some OpenMP regions contain areas of code that must be executed sequentially, such as calls to the MPI library.

individual node allows OpenMP to be used, while MPI is used to to communicate information between nodes. This suitability arises from the fact that MPI and OpenMP differ significantly in terms of the visibility of memory between cores. In MPI, each process has its own memory space, and the movement of data between processes is explicitly controlled using calls to the MPI application programming interface (API). In contrast, OpenMP is a shared memory system, and all threads spawned by a given process have access to the same memory. The relevance of hybrid approaches is currently increasing, as trends in processor development focus on increasing shared-memory parallelism (more cores per node), rather than faster processors. Simultaneously, the number of nodes available in state-of-the-art clusters is continuously increasing.

ONETEP[1] is a software package implementing linear-scaling density functional theory[8,11,14] to calculate total energies, forces, and a range of other properties of systems of hundreds up to tens of thousands of atoms.[15,16] The linear-scaling computational cost with respect to the number of atoms within ONETEP is achieved through the exploitation of the "near-sightedness of electronic matter" principle.[17,18] A unique feature of this code is that it can achieve linear-scaling while maintaining the high basis set accuracy of conventional plane-wave codes. The development of the hybrid MPI-OpenMP parallel scheme described in this article results in a significant improvement in the number of cores that can be efficiently utilized as well as lifting limitations the maximum number of cores that can be used per atom.

In Section 2, we describe the ONETEP package, focusing on the current approach to parallelism, then the implementation of the hybrid MPI-OpenMP approach. Section 3 shows and discusses the performance of the hybrid code performing single point energy calculations on various chemical systems ranging in size from hundreds to tens of thousands of atoms using different HPC facilities. Section 4 discusses an initial investigation into the mechanism by which sonification decomposes a cellulose nanofibril into its component layers. Calculations are made possible by the developments presented here. We finish with conclusions in Section 5.

2. ONETEP METHODOLOGY AND PARALLEL SCHEME

2.1. Methodology. The theoretical basis of the ONETEP methodology is discussed in detail elsewhere[1] and are only summarized here: The ONETEP program is based on a reformulation of DFT in terms of the one-particle density matrix, ρ ($\mathbf{r},\mathbf{r}'$), expressed in the following form

$$\rho(\mathbf{r}, \mathbf{r}') = \sum_{\alpha} \sum_{\beta} \phi_{\alpha}(\mathbf{r}) K^{\alpha\beta} \phi_{\beta}(\mathbf{r}') \tag{1}$$

where the "density kernel" $\mathbf{K}$ is the density matrix expressed in the duals of the set of nonorthogonal generalized Wannier functions (NGWFs)[19] $\{\phi_{\alpha}(\mathbf{r})\}$. The calculation of the electronic energy within ONETEP takes the form of two nested loops, the density kernel, $\mathbf{K}$, and NGWFs $\{\phi_{\alpha}(\mathbf{r})\}$ are optimized within the inner and outer loops, respectively. The NGWFs are constrained to be strictly localized within spherical regions centered on atoms, and their shape is optimized self-consistently by expressing them in a psinc basis set[20] which is equivalent to a plane-wave basis set.[21] As a result, ONETEP is able to achieve linear-scaling computational cost while retaining the large basis set accuracy characteristics of plane-wave codes.

The psinc functions used within ONETEP are centered on the points of a regular real-space grid and are related to a plane-wave basis through fast Fourier transforms (FFTs). In order to perform operations involving NGWFs with a computational effort independent of system size, the FFT box technique is used.[22] An FFT box is a box of grid points centered on the atom associated with an NGWF and large enough to contain any overlapping NGWF localization spheres in their entirety. This representation permits the use of plane-wave methodology to perform momentum space operations without the computational cost scaling up with the size of the simulation cell.

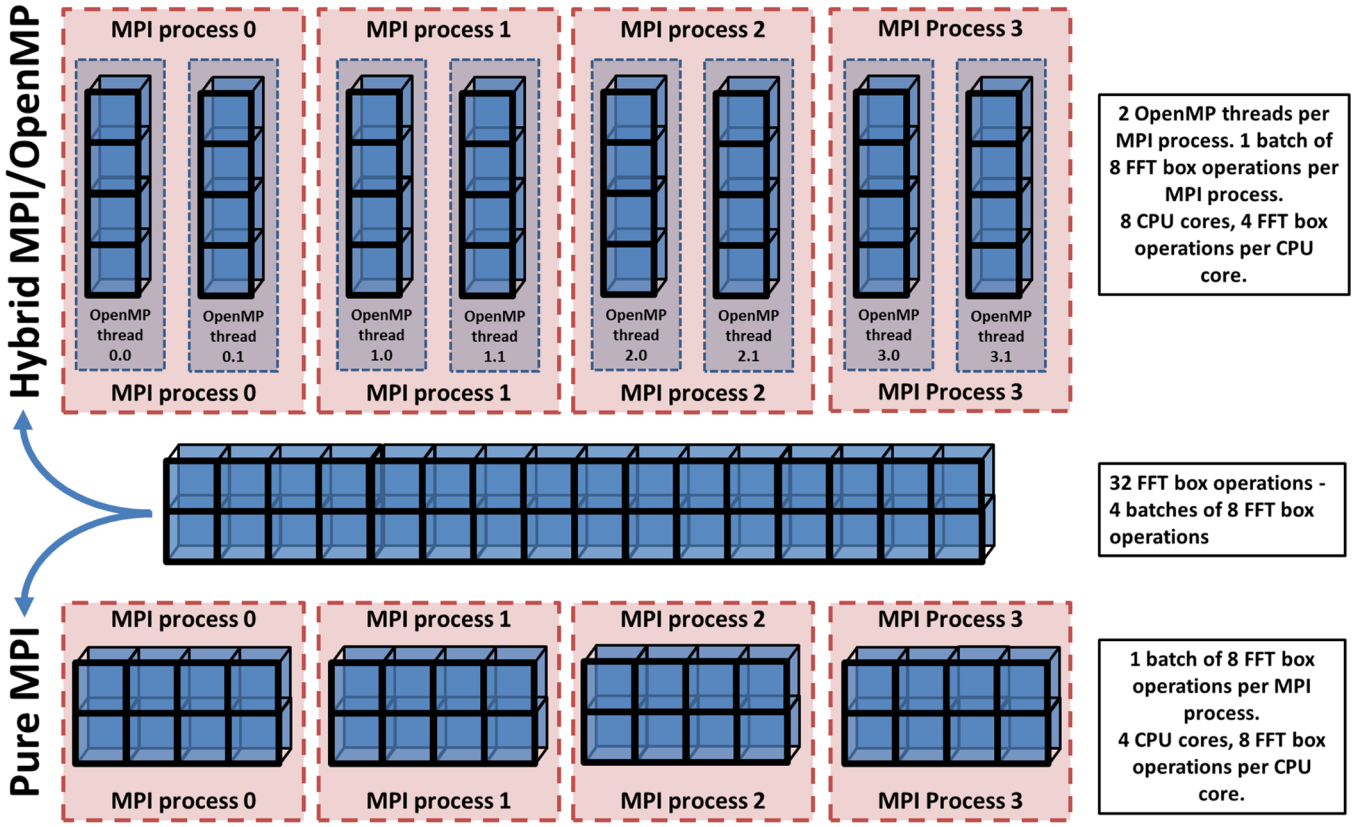


Figure 2. Distribution of FFT box operations across MPI processes and OpenMP threads in the hybrid MPI-OpenMP code.

The OpenMP paradigms utilized within the hybrid MPI-OpenMP version of ONETEP are illustrated in Figure 1. These paradigms are implemented through the addition of pragmas such as !$OMP PARALLEL and !$OMP END PARALLEL which are used to declare the start and end of regions of code executed in parallel.

2.2. The FFT Box Technique. Operations performed on FFT boxes represent one of the largest computational workloads in ONETEP. In particular, the calculation of the charge density, local potential integrals, and the NGWF gradient all employ these operations extensively. The routines used to calculate these properties also represent areas of the code with significant memory demands and communication overheads. The numerical details of the FFT box operations have been described in detail in an earlier paper[22] and more recently when we described the implementation of these algorithms on GPU based accelerators.[23] Here, we discuss the distribution of work and communication between cores in the hybrid code, as the numerical algorithms are unchanged.

At the MPI level of parallelism, the FFT box operations are not distributed directly. Rather, the overall workload associated with atomic data is distributed between MPI ranks using a space-filling curve to distribute atoms, weighted according to the number of NGWFs centered upon the atoms. In the hybrid code, the same space-filling curve is used to distribute the workload across MPI processes. However, during the calculation of properties that are constructed from data produced by FFT box operations, the FFT box operations associated with an MPI process are distributed evenly across OpenMP threads using a partitioning approach.

Figure 2 outlines the effect this new algorithm has on the distribution of the FFT box operation workload across CPU cores. The FFT box operations associated with NGWFs for an MPI process are divided into batches; in the case of the MPI-only code, a single core iterates over each member of the current batch, while in the hybrid version the members of the batch are distributed across a number of cores defined by the number of OpenMP threads.

FFT box operations generate data that is not typically stored across MPI processes in the atomistic manner defined by the space filling curve described above. For example, the charge density is stored as slabs of the simulation cell. This means that communication of the data produced by FFT box operations is necessary. During this stage of the process the master thread of each MPI process deals with the exchange of data with other MPI processes.

The use of the hybrid scheme means that the total number of cores that may be used can exceed the number of atoms, a hard limit within the MPI-only code. Further, the amount of memory used per core is reduced as several cores share the same NGWF data and simulation cell data. This is particularly useful on systems with a relatively low amount of RAM per core (less than 1 GB per core).

2.3. Sparse Algebra. Many of the operations in LS-DFT codes rely on iterative algorithms involving sparse matrix algebra. For example, approaches based on the Hotelling algorithm[24] are used to invert the overlap matrix; the canonical purification scheme of Palser and Manolopoulos,[25] or the CG-optimization scheme of Li, Nunes, and Vanderbilt[26] can be used to optimize a density matrix. In all cases, rapid matrix−matrix multiplications are required, on large matrices whose dimension is the number of localized orbitals, and whose fraction of nonzero elements is anywhere from under 1% up to 100%.

dx.doi.org/10.1021/ct500686r | J. Chem. Theory Comput. XXXX, XXX, XXX−XXX

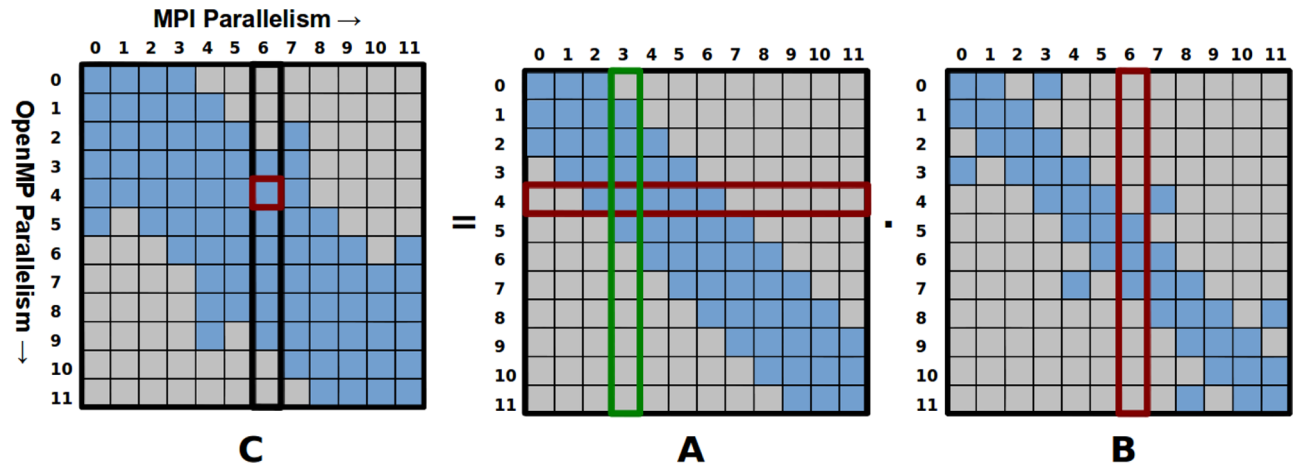


Figure 3. Schematic of the parallel decomposition of the workload for a sparse matrix multiplication under the hybrid OpenMP-MPI scheme, on 12 MPI processes. Blue shading indicates segments containing nonzero elements. The red boxes highlight a specific segment of **C** local to MPI process 6 and the range of segments of **A** and **B** which contribute to it. The green box indicates the set of segments communicated by MPI process 3, of which only some are nonzero. The OpenMP parallelism divides up the workload of each MPI process by dynamically distributing the segment pair matrix product operations between available threads.

Previous work[27,28] has described the work done to optimize the parallel performance of sparse matrix multiplications. Briefly, the algorithm is as follows: a typical operation is **C** = **A**·**B**, where **C**, **A**, and **B** are all sparse matrices of dimension such that the number of rows of **A** and **C**, the number of columns of **B** and **C** are equal, and the number of columns of **A** equals the number of rows **B**. The data associated with each matrix is divided by columns over the MPI processes according to the atom of each column function. The same scheme is used again to subdivide the range of rows within each process' data, resulting in segments. Each segment is represented internally in either a dense format (all elements stored), or a sparse format (only nonzero atom-blocks are stored), or is blank (no nonzero elements) as appropriate. Essentially, this is an atom blocked, column-indexed format unique to ONETEP but is closely related to the compressed column storage format. Segments $\mathbf{C}_{nm}$, labeled by row and column segment indices n,m which each run over the range of MPI processes, can be calculated by summing over the relevant segment−segment matrix products: $\mathbf{C}_{nm} = \sum_l \mathbf{A}_{nl}.\mathbf{B}_{lm}$. This necessitates the communication of the relevant segments of **A** from the process on which they are stored to any processes on which **C** requires them in the above summation. A "crop" is performed first, however, such that only **A** data which will actually contribute to the result is transmitted, as determined by the sparsity patterns of **B** and **C** (see Figure 3).

The parallel efficiency is further improved by grouping together subsets of the nodes and sharing their **B** data in advance, such that **A** data need only be transmitted to the same-ranked processes in other groups, with the resulting contributions to **C** being collated afterward over the processes in the group. Therefore, on any given process there is an outer loop over other processes n to receive data from, followed by an inner loop over pairs of segments $\mathbf{A}_{nl}$ and $\mathbf{B}_{lm}$. It is at this stage that we are able to introduce an extra level of parallelism with OpenMP. For each set of received segments of **A**, the loop over segment pairs is divided over the available OpenMP threads and individually dispatched as either a LAPACK dgemm call (for pairs of dense segments) or a series of individual sparse block multiplications. This OpenMP loop is subject to dynamic load-balancing as the computational effort of each segment pair product may vary significantly depending on the degree of sparsity of the segments.

The second level of parallelization thus created also further enables the communications workload to be hidden behind computation. The master thread of each MPI process deals with receiving, sorting, and cropping the index and data of **A** and replying to requests by other MPI processes for local segments of data and only processes actual multiplication of segment−segment pairs when no communication is currently required. All other threads work exclusively on the multiplication of segment−segment pairs. This is designed to ensure that by the time the other threads have dealt with their segment pairs, the next set of segments of **A** have already been received, ensuring there is no communications overhead from the perspective of the majority of the threads. For large systems with large kernel cutoffs (hence many nonzero elements in the matrices), switching from an MPI-only communications strategy to a hybrid OpenMP-MPI strategy generally produces a major efficiency gain in the sparse matrix algebra routines at high total core counts. This is because, at fixed total core count, as thread count rises and process count falls, the workload is being divided into larger chunks which can each individually be processed more efficiently with dense linear algebra.

2.4. Other. Apart from FFTs and matrix algebra, the final sizable fraction of the computational load in a ONETEP calculation consists of the operations on FFTbox data to either construct the row sum $\sum_\beta K^{\alpha\beta}\phi_\beta(\mathbf{r})$ or evaluate matrix elements of the form $\langle\phi_\alpha|\hat{O}|\phi_\beta\rangle$ for various operators $\hat{O}$. In these cases, the FFT box data is already efficiently parallelized over MPI processes according to the distribution of the atoms. The OpenMP threads can best be harnessed by allowing the various threads to work on an FFT box each. In the row sum operation, each NGWF $\phi_\beta(\mathbf{r})$ that overlaps any of the functions $\phi_\alpha(\mathbf{r})$ in the current batch of functions local to the process is considered in turn: an OpenMP thread is assigned to each FFTbox to which the NGWF $\phi_\beta(\mathbf{r})$ contributes and adds $K^{\alpha\beta}\phi_\beta(\mathbf{r})$ to each grid point $\mathbf{r}$.

Likewise, during matrix element calculations, for each ket function $\hat{O}|\phi_\beta\rangle$ in the batch, each overlapping NGWFs $\langle\phi_\alpha|$ is considered: an OpenMP loop extracts the data of $\hat{O}|\phi_\beta\rangle$ from the FFT box corresponding to the points in common between

dx.doi.org/10.1021/ct500686r | J. Chem. Theory Comput. XXXX, XXX, XXX−XXX

Table 1. Hardware Systems Used To Perform Benchmark Calculations

system	cores	nodes	cores per node	cores per NUMA region	hardware threads per core	theoretical performance (TFlop/s)
BlueJoule	114,688	7,168	16	16	4	1,426
ARCHER	72,192	3,008	24	12	2	1,560
Iridis 4	12,200	750	16	8	2	250

$\phi_\beta(\mathbf{r})$ and $\phi_\alpha(\mathbf{r})$ and then calculates $\langle\phi_\alpha|\hat{O}|\phi_\beta\rangle$. Because these operations each act on different FFTboxes, there is effectively no OpenMP synchronization overhead, and the scaling with thread count is efficient as long as the batches of NGWFs are large.

Another section of the code subject to OpenMP parallelism is the nonlinear-scaling Ewald summation,[29] which calculates the electrostatic interaction between the point charges representing the ion cores at positions $\mathbf{R}_I$, $\mathbf{R}_J$ and charges Z_I, Z_J. In terms of the Ewald potential V_{EW} and Madelung potential v_M for a given simulation cell, the expression to be evaluated has the general form

$$E_{\mathrm{II}} = \frac{1}{2}\sum_{I,J\neq I} Z_I Z_J (v_{\mathrm{EW}}(\mathbf{R}_I - \mathbf{R}_J) - v_M) \quad (2)$$

Here, the inner loop over atom pairs IJ can be parallelized with OpenMP (the outer loop already being parallelized with MPI). Near-perfect scaling with both OpenMP threads and MPI processes can be achieved as long as the thread and process count both remain significantly smaller than the number of atoms. It is of note that the current implementation of the Ewald summation in ONETEP is not formally linear-scaling. In the calculations that are routinely performed using ONETEP this has not been an issue as the overhead relating to this area of the code is small - for example in the applications on cellulose that we present in Section 4, for our largest system with 21,276 atoms the Ewald takes 3% of the total time in each single point energy calculation which consists of 9 NGWF iterations.

Finally, in all sections of the code dealing directly with grid data on whole-cell grids, such as evaluation of the exchange-correlation energy and potential, there is a further opportunity for thread-parallelism. The data distribution (as described in ref 30) of realspace data is in slabs perpendicular to the third lattice vector the simulation cell (the '3' direction, typically along z). Each MPI process allocates memory and performs computations only for the points $\mathbf{r}_i$ local to its slab. Within each MPI process, the loop over the grid points i within the local slabs can be OpenMP-parallelized very efficiently, for example when evaluating the exchange correlation energy $\varepsilon_{xc}(n(\mathbf{r}))$.

3. BENCHMARKS

For measuring the performance of a large scale parallel code the most important quantity from an application perspective is the actual time to solution. Therefore, for most users, the most appropriate metric is the "Strong scaling", describing the relative performance of the code as the number of cores are increased for a given problem size. Ideal (linear) strong scaling is often difficult to achieve since as the number of cores increase, the total volume of communication of data between cores, and the number of individual messages, also increases. In addition, parts of the code, such as control logic, may not be amenable to parallelization,[31] particularly if there are two levels of parallelization as in the current MPI-OpenMP hybrid scheme. A second metric is the "weak scaling" which describes the behavior of the time to solution as both problem size and the number of cores used are simultaneously increased. A further metric that may be used is the parallel efficiency (PE), which gives the speed-up obtained on a given number of cores relative to the ideal speed-up from a chosen reference number of cores, thus giving a measure of the actual performance in relation to the optimal performance. We employ a rule-of-thumb in this work that a PE of less than 0.5 represents a wasteful use of resources.

Significant effort has been devoted to achieving good weak parallel scaling of LS-DFT codes so as to demonstrate the feasibility of very large calculations. For example, the CONQUEST LS-DFT code has been used to perform calculations on 2,097,152 bulk silicon atoms using 4096 CPU cores,[32] which is equivalent to 512 atoms per core. It has also been shown to exhibit very good weak-scaling from 8 to 4096 cores for the bulk silicon system mentioned above and good strong-scaling between 16 and 128 cores for hut clusters of Ge on the Si(100) surface with 11,620 or 22,746 atoms. Other LS-DFT codes such as SIESTA[3] and FREEON[33] exhibit good scaling in the regime of thousands of atoms simulated on up hundreds of cores.[34]

In ONETEP, Hine et al. showed nearly ideal weak-scaling behavior on total energy calculations on strands of model DNA systems from 2091 to 16775 atoms on 32 to 256 cores and respectable strong-scaling to 256 cores.[28] Since then, further unpublished developments improved the limit of high-efficiency strong-scaling with MPI process count (as defined by keeping the parallel efficiency greater than 0.5) to over 1024 processes, at least for large systems. In the work described here, however, we are able to demonstrate that significantly larger core counts can be reached without loss of parallel efficiency.

In the following, we have carried out benchmark calculations on a number of systems: the Iridis 4 supercomputer at the University of Southampton; the UK's national supercomputer, ARCHER, a Cray XC30, at the University of Edinburgh; and BlueJoule, an IBM BlueGene/Q system at the STFC's Hartree Centre. The specifications of these machines are outlined in Table 1. They were chosen for this study as they are representative of the range of large scale computing resources available to research groups in both academia and industry.

CPU cores on a node generally have access to all the shared memory on the node. However, in "non-uniform memory access" (NUMA) systems the memory access times may depend on the location of the memory relative to the core. The Iridis 4 and ARCHER both contain 2 CPU sockets per node. This gives rise to NUMA issues as the two sockets are on different PCI express buses along with half of the nodes RAM. As such, in order for a core on one bus to retrieve data from a memory location in the RAM on the other bus it must transfer data across the quick path interconnect bus, which is significantly slower than the PCI express bus, resulting in reduced access speeds. A second NUMA issue arises on machines such as the recently discontinued HECToR supercomputer, where a single, 16 core, processor contains two processor dies, each with its own L3 cache and 8 CPU cores. In

contrast, BlueGene/Q systems such as BlueJoule are truly symmetric multiprocessor systems where all cores have equal access to the memory on a node. Memory-locality must be considered carefully when executing codes on machines with NUMA. In practice we find that on all systems tested here, it is highly detrimental to spread the threads of a given MPI process over more than one NUMA region. For example, on ARCHER, a calculation performed on an ATC trinucleotide containing 189 atoms using production-quality settings takes 2,162 s when the threads associated with an MPI rank are split across 2 NUMA regions but 1,389 s if they are restricted to a single NUMA node.

Many modern CPUs support simultaneous multithreading (SMT) a form of hardware multithreading in which a single CPU core executes code for multiple software threads. The use of SMT was found to be detrimental to performance in the case of the FFT box operations within ONETEP on Intel cores. This is thought to be caused by either competition between threads for limited memory bandwidth or over subscription of the available shared cache resources in the FFT regions of the code. This is a common problem with memory-bound algorithms such as FFTs. In contrast, the sparse matrix code, consisting of many small matrix−matrix multiplications, does not face such limitations, and the use of SMT results in a benefit to performance.

3.1. Small Systems. Initial tests were performed using a small system, an ATC trinucleotide containing 189 atoms. These calculations were run for 1 NGWF iteration with production-quality settings (no kernel threshold, 8.0 a_0 NGWF radii, 800 eV psinc kinetic energy cutoff). In the previous MPI-only implementation, the number of cores that could be used in such a calculation was limited by the need to retain at least 1 atom per MPI process, so here we demonstrate that the new hybrid implementation is able to significantly exceed this limit while retaining favorable scaling. Indeed, we show that it is possible to exceed 1 core per NGWF.

Figure 4(a) (and inset) shows the total time for an electronic energy minimization on the Iridis 4 supercomputer; the best performance that can be achieved using the MPI-only implementation is highlighted in the inset. Panel (b) shows the strong scaling performance, and panel (c) shows the parallel efficiency. For each total number of cores, separate curves are shown for different numbers of OpenMP threads per MPI process.

It is seen that the performance of calculations with a larger number of OpenMP threads per MPI process improves as the total number of cores used increases. For example, when using 64 cores in total the best performance is achieved when using 2 OpenMP threads per MPI process, while at 512 cores in total, optimal performance is obtained using 8 threads per MPI process. This finding has implications for the optimal manner in which calculations should be submitted to a machine as calculations with smaller total core counts may benefit from the use of the MPI-only approach.

Parallel efficiency is seen to be greater than 0.4 up to 256 cores in the hybrid OpenMP-MPI runs, and the total time to solution can be reduced by a factor of 2× relative to the MPI-only implementation. It should be emphasized that this is a rather smaller system than typically run with ONETEP and is included here principally to demonstrate that the speedup obtained through hybrid parallelism is not limited to large systems.

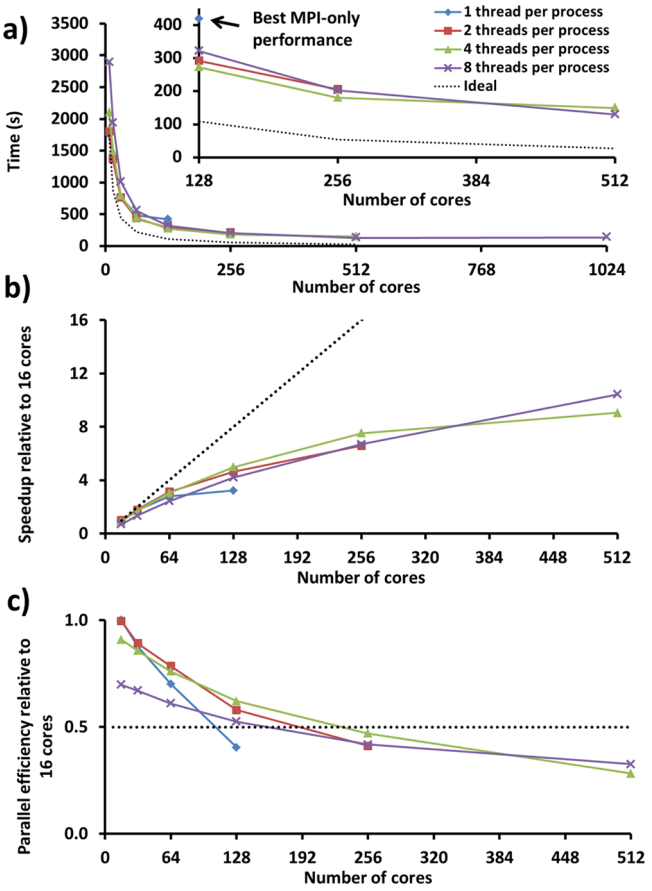


Figure 4. Scaling of the hybrid MPI-OpenMP code for calculations performed on a 189 atom nucleotide using the Iridis 4 supercomputer, for varying choices of number of OpenMP threads per MPI process. The inset in panel a) shows the results for larger core counts in more detail.

3.2. Intermediate Systems. We next present the benefit that is afforded to more typical problems, containing thousands of atoms. Figures 5 and 6 show the performance of the hybrid code for production-quality calculations on a series of systems containing approximately 4000 atoms (a zigzag carbon nanotube (CNT) using a 50.0 a_0 density kernel cutoff, 8.0 a_0 NGWF radii, and 800 eV psinc kinetic energy cutoff; a H-terminated GaAs nanorod using no density kernel truncation, 9.0 a_0 NGWF radii, and 700 eV psinc kinetic energy cutoff; a supercell of bulk silicon using no density kernel truncation, 8.0 a_0 NGWF radii, and 800 eV psinc kinetic energy cutoff; a strand of B-DNA comprising a sequence of 64 randomized base pairs using 100.0 a_0 density kernel cutoff, 8.0 a_0 NGWF radii, and 800 eV psinc kinetic energy cutoff). These calculations were run for 1 NGWF iteration, and all simulations were performed on the ARCHER supercomputer. Specifically, Figures 5 and 6 show parallel efficiency and speed-up, respectively. These values are measured relative to the same runs on just 48 cores with the MPI-only code, which is the smallest number for which all the systems fit in the available memory. We use 5 different balances of MPI vs OpenMP, ranging from 24 MPI processes per node each with 1 thread, up to just 2 MPI processes per node, each with 12 threads. For each of these choices of balance, we use a total number of cores ranging from 240 to 3840.

An MPI-only strategy is seen to give satisfactory performance (parallel efficiency greater than 0.5) only up to about 1000

dx.doi.org/10.1021/ct500686r | J. Chem. Theory Comput. XXXX, XXX, XXX−XXX

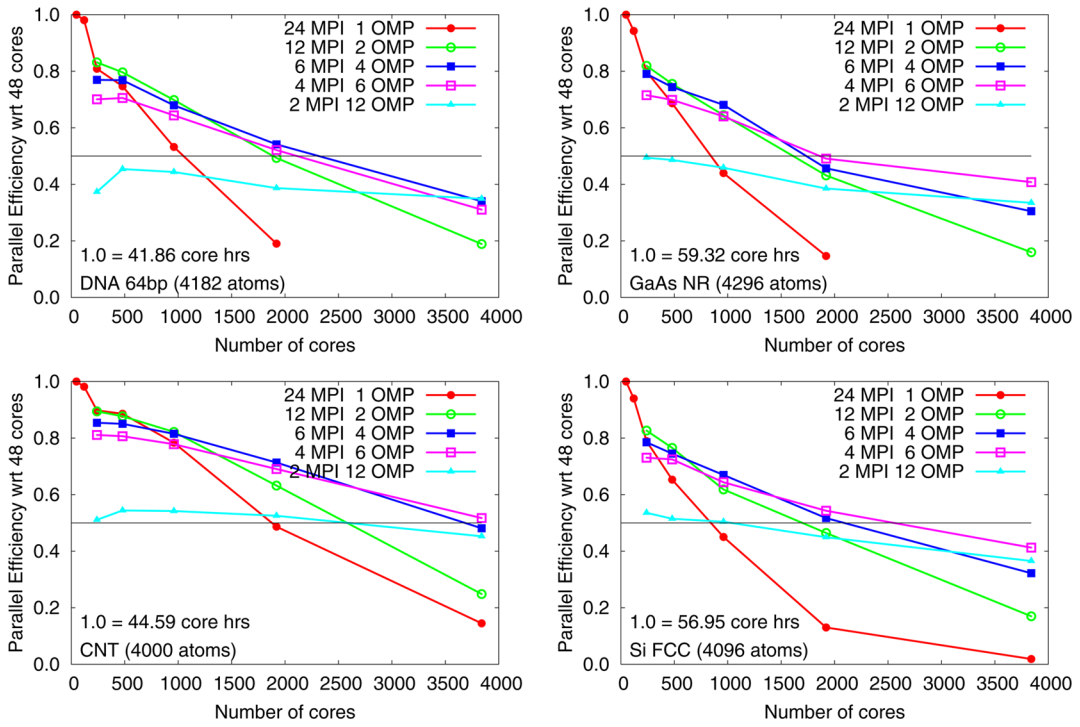


Figure 5. Parallel efficiency of a variety of chemical systems containing approximately 4000 atoms on ARCHER. Calculations were run for 1 NGWF iteration with production-quality settings.

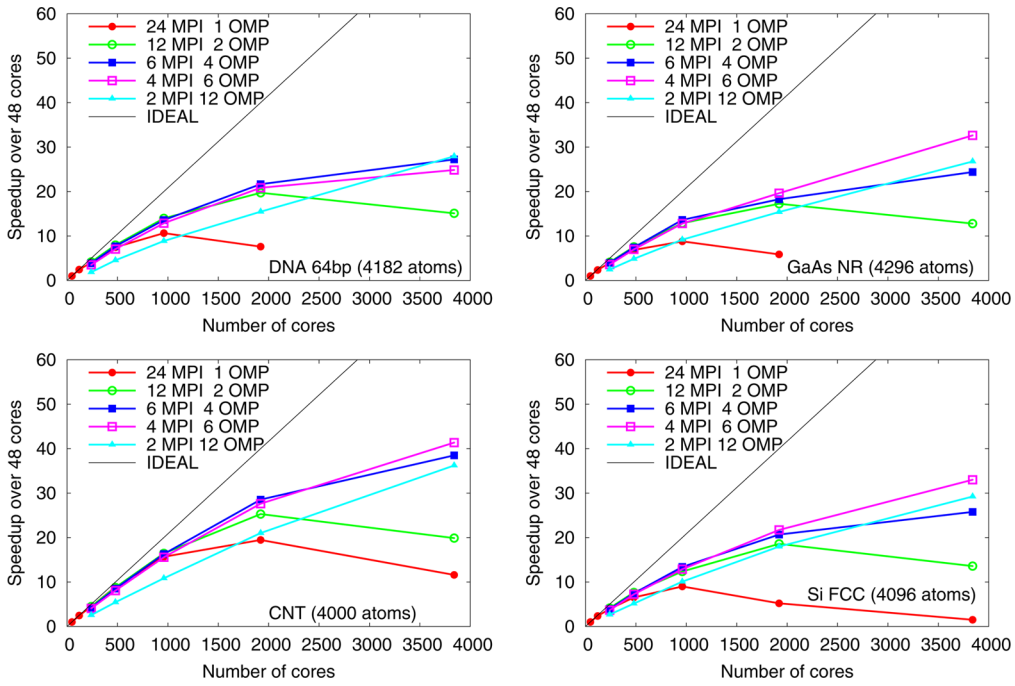


Figure 6. Strong scaling of a variety of chemical systems containing approximately 4000 atoms on ARCHER. Calculations were run for 1 NGWF iteration with production-quality settings.

cores. Increasing the level of OpenMP multithreading again incurs a parallel efficiency hit for each extra thread, so at small core counts it is not beneficial. However, crucially, it preserves the scaling to much higher total core counts without nearly as much loss of parallel efficiency. For example, a run with 8 threads per process has almost the same MPI efficiency at 4096 cores as an MPI-only run at 512 cores. Therefore, the hybrid code gradually becomes much more favorable at higher core counts. For 6 OpenMP threads per process, a parallel efficiency of around 0.5 compared to a 48 core run is retained even at 4096 total cores (i.e. at around one core per atom). This significantly reduces the wall time in which such jobs can be completed. For example, for the CNT, the maximum number of cores on which the job could be run with PE > 0.5 was 960, whereby each NGWF optimization iteration took 213 s. Contrastingly, with 6 threads per MPI process on 3840 cores

dx.doi.org/10.1021/ct500686r | J. Chem. Theory Comput. XXXX, XXX, XXX−XXX

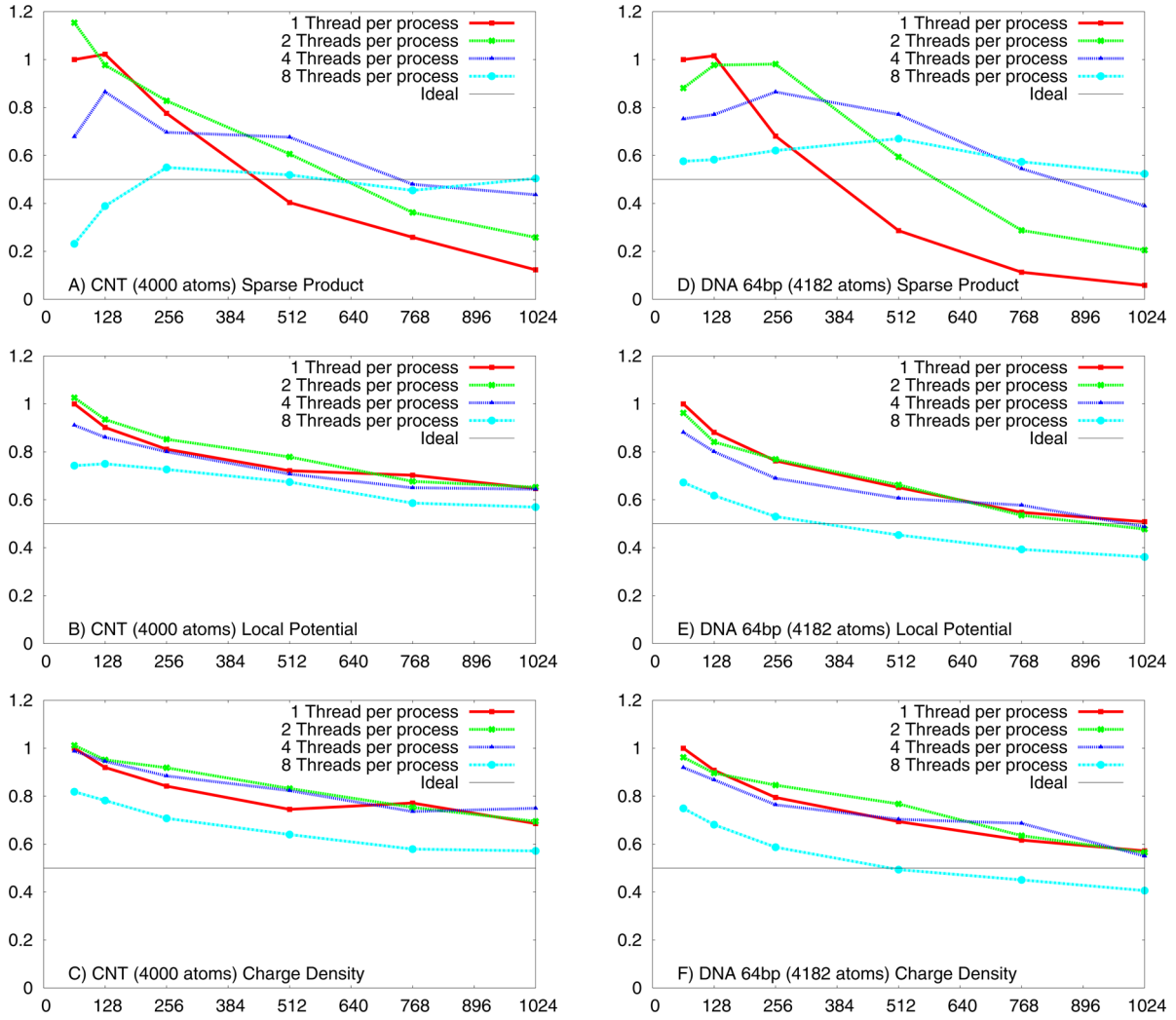


Figure 7. Parallel efficiencies for timings of individual routines considered in isolation from the rest of the code. The P.E. for charge density (C,F), local potential (B,E), and sparse matrix products (A,D) sections of the code are shown, for calculations performed on zigzag carbon nanotube (CNT) (left) and a strand of B-DNA comprising a sequence of 64 randomized base pairs (right). Calculations were performed on Iridis 4 and run for 1 NGWF iteration with production-quality settings.

this is reduced to 81 s, a speed-up of 2.6×. Comparison to Figure 4 shows that slightly more cores per atom may be used for the larger systems before parallel efficiency is lost (between 1 and 2 cores per atom for the 4000 atom systems and 1 core per atom for the smaller system). This occurs because the number of FFT box operations per atom increases with size of the system, while the number of NGWF overlaps approaches its asymptotic value that increases linearly with the number of atoms.

Figure 7 shows the parallel efficiency of the sections of the code used in the calculation of the charge density, the local potential integrals, and the products of sparse matrices - the major bottlenecks in calculations of this size. Calculations on a zigzag carbon nanotube (CNT) and a strand of B-DNA comprising a sequence of 64 randomized base pairs performed on the Iridis 4 supercomputer were used for this analysis.

Figure 7 attempts to break down the overall parallel efficiency according to the various computational tasks performed during a typical run, focusing on evaluation of the charge density, evaluation of matrix elements of the local potential in the NGWF basis, and on sparse matrix algebra.

The FFT box operations demonstrate uniform scaling with a number of MPI processes and incur a P.E. hit with each added thread: this is likely to be due to the thread-serialization during the operations involving whole-cell grids. Meanwhile, in the sparse matrix product operations, because the total communications volume increases with MPI process count, the scaling with number of MPI processes is relatively poor and each increase in thread count improves the scaling with total core count. The overall efficiency results from the balance of these two components, which at large system sizes are close to equal parts of the total load. Future work should therefore be aimed at improving the thread-parallelization of the whole-cell grid operations and the MPI-parallelization of the sparse matrix algebra.

3.3. Large Systems. Finally, we demonstrate the ability to simulate very large systems, containing tens of thousands of atoms on BlueJoule, a Tier 0 computational resource currently ranked 23rd in the Top500 list of supercomputers.

The BlueJoule platform allows us to take advantage of SMT for suitable areas of code. In the following benchmarks we have used SMT for all operations except the FFT box operations. As such, the number of OpenMP threads per MPI process are

dx.doi.org/10.1021/ct500686r | J. Chem. Theory Comput. XXXX, XXX, XXX−XXX

denoted as "4/16" to indicate 4 threads for the cache intensive FFT box operations and 16 threads for other areas of the code.

As the amount of RAM per core on BlueJoule is relatively low, it is not possible to utilize every CPU core on a node when using the MPI-only implementation. This necessitates the use of a "number of cores charged" metric as these unused cores are still classed as active in such a calculation.

When running at this scale we observe that the balance of computational effort has shifted considerably compared to the smaller systems, and sparse matrix algebra is now the largest individual component. Because these routines begin to exhibit poor scaling with MPI process count beyond a few thousand processes, due to the volume of point-to-point communications, greater levels of OpenMP threading and correspondingly smaller MPI process counts are seen to considerably improve the performance.

Figure 8(a) shows the performance of the hybrid MPI-OpenMP code for a 13,969 atom beta amyloid fibril protein,

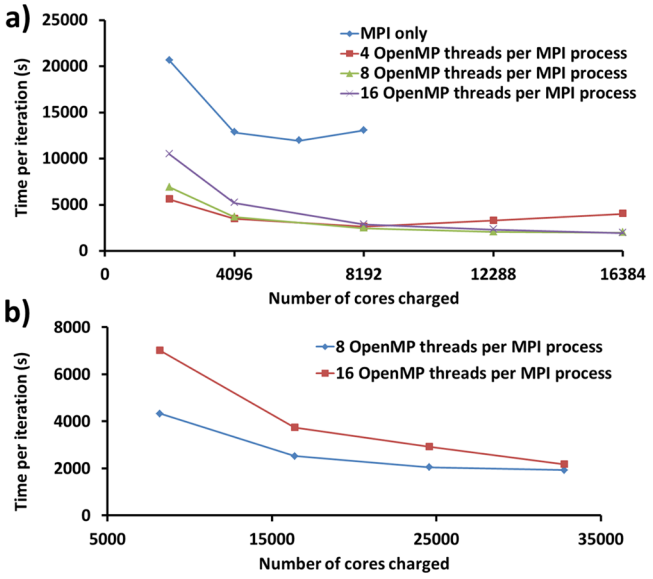


Figure 8. a) Total time for 1 iteration of the 13,969 atom beta-amyloid fibril, for MPI-only (blue), and 4/16, 8/32, and 16/64 OpenMP threads per MPI process (red, green, and purple, respectively). b) Total time for the 41,907 atom Amyloid fibril trimer. Both sets of calculations consisted of 1 iteration of the NGWF optimization loop with production-quality settings (5 iterations of the density kernel loop, 30.0 a_0 density kernel cutoff, 8.0 a_0 NGWF radii, 800 eV psinc kinetic energy cutoff).

run on the BlueGene/Q. These calculations were run for 1 iteration of NGWF optimization with production-quality settings (5 iterations of the density kernel loop, 30.0 a_0 density kernel cutoff, 8.0 a_0 NGWF radii, 800 eV psinc kinetic energy cutoff). A full single point energy calculation requires 10−15 such NGWF iterations.

Figure 8(a) shows that the use of MPI alone scales to only 2048 active cores (4096 cores charged). Contrastingly, the use of the hybrid code allows us to increase this by at least a factor of 4, to 16,384 cores. There is a corresponding increase in performance of up to 6× when compared against the fastest MPI-only calculation. As observed for the smaller systems described above, the performance of the hybrid code depends strongly upon the number of OpenMP threads per MPI process. The MPI-only test is the slowest throughout and, as mentioned above, does not scale to the largest number of cores. At lower total core counts it is seen that 4/16 OpenMP threads per MPI process gives the best performance and 16/64 OpenMP threads per MPI process is still less efficient. This pattern reverses at higher core counts, presumably due to the reduction in MPI communication that results from the shift from MPI to OpenMP parallelism.

Figure 8(b) shows the performance of the hybrid MPI-OpenMP code for an even larger system, a beta amyloid fibril trimer, containing 41,907 atoms. These calculations were run for 1 NGWF iteration with the same settings as used for the 13,969 atom system described above. In this case, the high total memory demands mean that the system was run only for 8/32 and 16/64 OpenMP threads per MPI process, from 8,192 to 32,768 cores. We observe that the performance of the 8/32-thread version surpasses that of the 16/64-thread version although the difference in performance reduces as overall core count is increased.

It is instructive to compare the performance of the current Hybrid OpenMP-MPI implementation with other similar codes where possible. However, such comparisons are hard to make fair, since different approaches to LS-DFT make very different approximations in relation to underlying basis sets and their approaches to localization and truncation of sparse matrices. These factors make it difficult to judge when comparable accuracy is achieved in two different approaches. In addition, fair comparisons must be run on equivalent hardware and with both codes set up to achieve optimal performance on that hardware, preferably by their respective developers. Such a detailed comparison is therefore beyond the remit of the current article.

Nevertheless, we can make some preliminary comparisons on the basis of already-published work, particularly that by the developers of CP2K, who have applied hybrid parallelism to their code,[35] and in particular to the sparse matrix algebra with the DBSCR library.[36] Several fundamental differences exist between these codes: first, CP2K uses Gaussian basis sets augmented with plane-waves, and thus a larger number of functions per atom than the number typically used by ONETEP, where a minimal number of in situ optimized functions are used; second, CP2K applies thresholding to determine the truncation of its representation of the density kernel matrix, in contrast to the approach used in ONETEP, where all nonzero elements of intermediate matrices are retained until the final result of an optimization step is used to update the current kernel. Combined, these two facts mean that for the same system, matrices in ONETEP would be less sparse but be smaller overall. An estimate of the number of floating point operations of useful computational work (FLOPS) per second for sparse matrix operations can be made according to

$$\mathrm{FLOPS} = N^3 \times nz^2 \tag{3}$$

Here, N is the size of the sparse matrix, and nz is the fraction of nonzero elements, or occupancy. In ONETEP these estimates contain some unavoidable uncertainty because of the variation in sparsity levels arising from the use of varied, fixed sparsity patterns in different circumstances, as described above.

For example, for a 13,846-atom model of an amorphous organic hole conducting material in ref 36, a 133,214 × 133,214 matrix is obtained with 18% occupancy, whereas in this work (Figure 8), the closest comparable system is the 13,969 atom Amyloid Fibril, where the matrices are of size 36,352 × 36,352 and vary from 14% up to 60% nonzero elements. On a Cray

dx.doi.org/10.1021/ct500686r | J. Chem. Theory Comput. XXXX, XXX, XXX−XXX

XC30 architecture, the DBSCR library obtained average timings for sparse matrix product operations for the aforementioned matrices of 0.5 s at 32768 cores (153 teraFLOPS/s), ∼1.1 s at 8192 cores (70 teraFLOPS/s), and 12.0 s at 512 cores (6 teraFLOPS/s), indicating relative parallel efficiencies of 0.38, 0.68, and 1, respectively.

The closest comparison we could achieve for ONETEP on comparable XC30 cores was to run on 8160 cores and 528 cores (680 MPI processes × 12 threads and 44 MPI processes × 12 threads, respectively). We provide teraFLOP/s figures based on the range of sparsity patterns noted above, noting that denser matrices dominate the computational effort. At 8160 cores, average time per matrix multiplication was 0.59 s (1.6 to 19 teraFLOPS/s), and at 528 cores it was 5.95 s (0.1 to 2.9 teraFLOPS/s). This corresponds to a relative parallel efficiency of 0.65. Comparison with Figure 8 suggests that scaling up to 16384 cores would generate further reduction in computational time. For equivalent-sized physical systems on equivalent hardware, typical sparse matrix product operations in ONETEP therefore display comparable parallel efficiency scaling, and similar total times, but are still rather less efficient overall on the basis of total teraFLOPS/s of useful work, because the typical matrix sizes for CP2K are larger. This suggests there is room for further improvement in the sparse algebra routines in ONETEP in future work, despite the advances presented here.

4. SIMULATIONS OF LARGE CELLULOSE NANOFIBRILS

Cellulose is one of the most abundant forms of biomass, with a huge number of applications ranging from use as an energy source[37,38] to a support in tissue engineering.[39] Cellulose is a polymer of cellobiose monomers and is commonly found in the form of cellulose Iβ nanofibrils within plant cell walls. Cellulose Iβ nanofibrils are composed of a number of elementary cellulose chains, the exact number of chains depending upon the biological source. Figure 9a shows the 8-layer, 36-chain structure for cellulose Iβ derived from Ding and Himmel's model for primary maize cell walls,[40] and Figure 9b shows a truncated version of this model in which the number of elementary fibrils has been reduced.

Renneckar et al.[41,42] recently described experimental studies showing the fragmentation of cellulose nanofibrils along the sheets defined by the pyranose rings into mono- and bilayer structures after undergoing sonication. A molecular sheet delamination mechanism was proposed for this process wherein a single layer at a time was separated from the nanofibril. These experimental results are consistent with a structure wherein the intrasheet interactions are stronger than the intersheet interactions. A number of computational studies of cellulose Iβ have investigated the relative strengths of these interactions. In agreement with the experimental results, studies using plane-wave methods[43] suggested that the intrasheet interactions in cellulose Iβ are eight times larger than the intersheet interactions. However, this study used a unit cell containing just 2 cellobiose molecules and periodic boundary conditions, meaning the system was treated as fully crystalline and the finite size and exposed surface of a true nanofibril were not accounted for. A related study using cellulose Iα models and fragment molecular orbital Møller−Plesset second order perturbation theory (FMO-MP2) methods[44] show the intersheet interactions to be stronger than the intrasheet interactions for a truncated crystalline structure composed of 12 elementary chains 6 cellobiose units in length. However, the difference in energies was observed to reduce if the same structure was equilibrated using molecular dynamics methods.

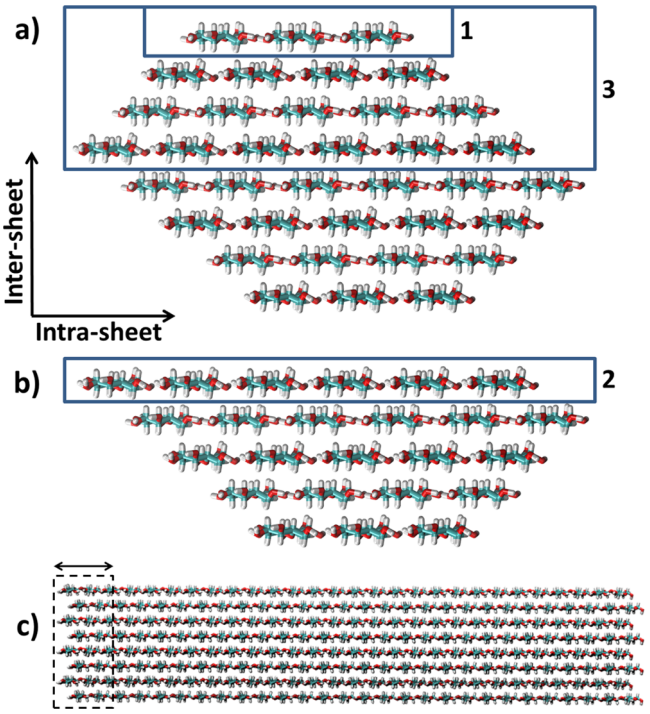


Figure 9. a) Axial view of a cellulose Iβ nanofibril containing 36 elementary chains. Chains contain either 1 or 14 cellobiose units each (1,620 and 21,276 atoms, respectively). b) Axial view of a truncated cellulose Iβ nanofibril containing 18 elementary chains. Chains contain 1 or 14 cellobiose units each (810 and 10,638 atoms, respectively). c) Cellulose Iβ nanofibril containing 36 elementary chains that are 14 cellobiose units in length. The length of a nanofibril containing a single cellobiose residue is highlighted for comparison. 1) Highlights the top layer of a 36 chain nanofibril which is translated in the 36 chain, 1 layer series of calculations, 2) Highlights the top layer of a truncated 18 chain nanofibril (18 chain, 1 layer) and 3) Highlights the top 4 layers of a 36 chain nanofibril which are translated in the 36 chain, 4 layers series of calculations.

Here, we illustrate that with the recent hybrid MPI-OpenMP developments in ONETEP we are able to perform investigations on larger, and much more realistic models of cellulose nanofibrils which capture their complex properties and move away from the heavily truncated models used in previous computational studies. Our aim here is to perform a preliminary study of the delamination process which is of fundamental importance for the structural implementations of cellulose nanofibrils in applications such as tissue engineering[39] and within a wide range of composite materials.[45] As this is a preliminary study, the effects of solvation and structural relaxation are beyond the scope of this paper, and we expect to investigate these effects in future work.

Cellulose Iβ nanofibrils, containing 36 elementary cellulose chains of 1 and 14 cellobiose units in length (Figure 9a/c), were constructed using the cellulosebuilder[46] software package. This structure was modified to create the truncated structure shown in Figure 9b by deletion of the relevant chains. Figure 9c emphasizes the differences between nanofibril models that are 1 and 14 cellobiose units in length. A nanofibril containing 14 cellobiose units was chosen for the large system here as it represents a system that could be used in a more complex future study that would examine the interactions between

dx.doi.org/10.1021/ct500686r | J. Chem. Theory Comput. XXXX, XXX, XXX−XXX

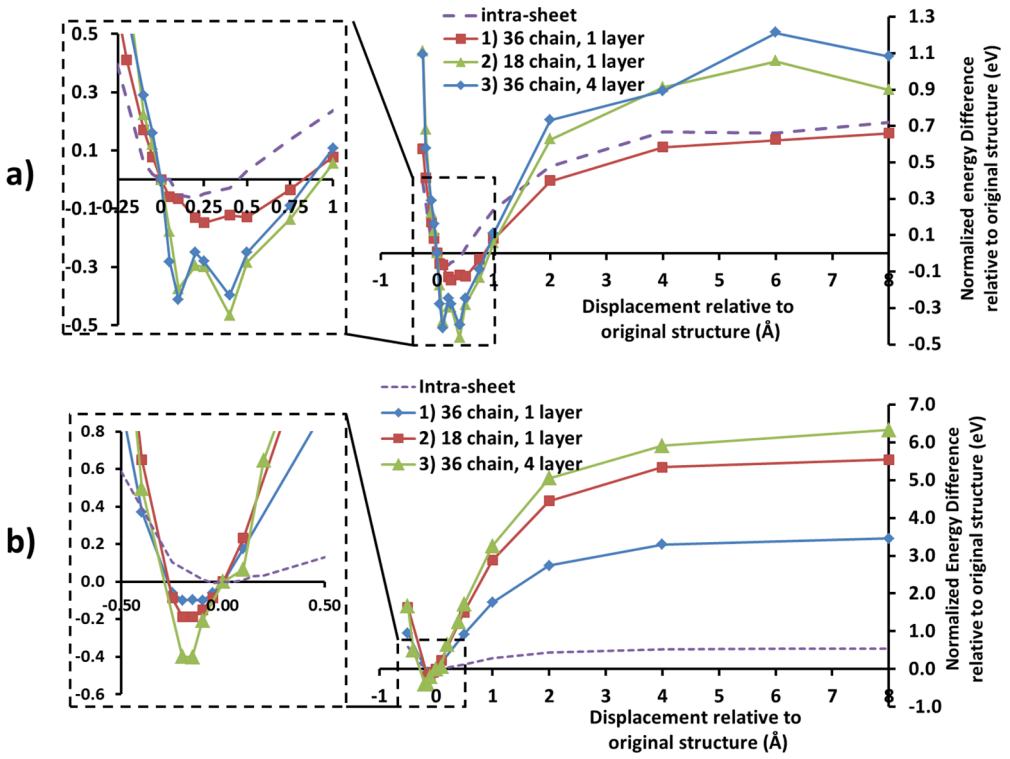


Figure 10. Interaction energy curves for cellulose nanofibrils 1 (a) and 14 (b) cellobiose units in length. Displacements are relative to the structure generated by cellulosebuilder. Data is normalized by the number of cellobiose units in each elementary chain.

Table 2. Interaction Energies for the Inter- and Intraplane Interactions within Model Cellulose Iβ Nanofibrils 1 and 14 Cellobiose Units in Length[a]

	normalized interaction energies for fibril 1 cellobiose unit in length (eV)	normalized interaction energies for fibril 14 cellobiose units in length (eV)
intrasheet	0.78	0.59
1) 36 chain, 1 layer	0.81	3.56
2) 18 chain, 1 layer	1.52	5.73
3) 36 chain, 4 layers	1.63	6.73

[a]Data is normalized by dividing by the number of cellobiose units in an elementary chain.

cellulose nanofibrils and other biological molecules such as cellulase enzymes which are used in the production of cellulosic ethanol.[37,38] A system of this size would be required in order to fully describe all of the interactions of a nanofibril with these enzymes.

In order to simulate potential energy curves approximating the delamination process, a series of structures were generated by translating single sheets (1 and 2), and groups of sheets (3), of elementary chains in the intersheet axis shown in Figure 9. To model the intrasheet interactions, a single fibril from a sheet containing 3 elementary fibrils (equivalent to that marked 1 in Figure 9a) was translated in the intrasheet axis shown in Figure 9. All single point energies were carried out using a 40.0 a_0 density kernel cutoff, 7.0 a_0 NGWF radii, and 800 eV psinc kinetic energy cutoff; all atoms had 4 NGWFs, except hydrogen atoms which had 1. The Grimme D2 correction[47] was used to model the van der Waals interactions. These calculations were run until full self-consistent convergence was reached with a NGWF gradient threshold of 2×10^{-6} $E_h a_0^{3/2}$. Simulations for the nanofibrils containing 14 cellobiose units per elementary chain were performed on BlueJoule, and those containing 1 cellobiose unit per elementary chain were performed on Iridis 4.

Figures 10a and b show the interaction energy profiles for nanofibrils that are 1 and 14 cellobiose units in length, respectively. Interaction energies calculated by taking the difference between the global maximum and minimum for each system are given in Table 2. Data is normalized by the number of cellobiose units in an elementary chain in both cases - the values for the system 14 cellobiose units in length were divided by 14 and those for the smaller system by 1.

Figure 10 shows that there is a significant difference in both the locations of the minima and the shapes of the curve when comparing systems of different sizes. In the larger system smooth curves with a minimum shifted slightly toward a negative displacement (−0.1 Å, reduced interplanar distance) are observed. In contrast, the curves for the small system show a minimum shifted toward a larger separation (+0.3 Å, increased interplanar distance) between layers, and multiple minima are observed. The results for the small system are qualitatively similar to those observed elsewhere[43] and may be explained by the sensitivity of the energy to individual van der Waals interactions in these smaller systems.

The intrasheet interaction energy per cellobiose unit appears to be stronger for the smaller system (0.78 vs 0.59 eV), despite the much larger number of favorable interactions in the larger system. This indicates that the intrasheet interactions are not

dx.doi.org/10.1021/ct500686r | J. Chem. Theory Comput. XXXX, XXX, XXX−XXX

simply additive and is thought to occur due to the presence of intrachain hydrogen bonds between adjacent cellobiose units that are present in the structure containing 14 cellobiose units per elementary chain but not in the smaller system containing a single cellobiose unit per elementary chain. The intrachain hydrogen bonds between different cellobiose units are highlighted "IC" in Figure 11; intrachain hydrogen bonds present in chains containing only a single cellobiose unit are not highlighted.

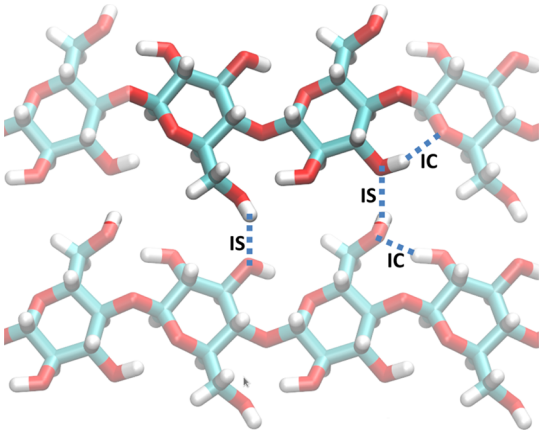


Figure 11. A cellobiose unit is highlighted as part of a cellulose chain. Intrachain (IC) hydrogen bonds are shown. We can also see a second chain which is connected to the first one via intrasheet (IS) hydrogen bonds. IC hydrogen bonds within a single cellobiose unit are also present but are not shown here.

For both the short and long nanofibrils it is observed that the interaction energies for the 36 chain, 4 layers (3) and 18 chain, 1 layer (2) systems are greater than that of the 36 chain, 1 layer system (1) ($1.52 \approx 1.63 > 0.81$ eV for the small system and $6.73 \approx 5.79 > 3.56$ eV for the large system). This is unsurprising as a greater number of elementary chains are translated in these systems, and thus a greater number of favorable interactions are broken. When considering the effect of truncation, the difference in energies between the 36 chain, 4 layers (3) and 18 chain, 1 layer (2) systems is small in both cases (0.10 eV for the small system and 1.00 eV for the large system). These differences show that the truncation of a cellulose nanofibril through the removal of sheets of elementary cellulose chains has an impact on the properties of the nanofibril.

The results for the smaller system indicate that the inter- and intrasheet interaction energies per cellobiose unit are similar (largest intersheet interaction - intrasheet interaction = 0.84 eV), while the larger system shows intrasheet interactions to be significantly weaker than the intersheet interactions (largest intersheet interaction - intrasheet interaction = 6.14 eV). This demonstrates the complexity of the system as experimentally it is known that sonication in water separates the sheets but does not separate the chains within a sheet. This is still consistent with our findings as the intersheet interactions are electrostatic and van der Waals driven and can thus be easily disrupted at the individual level by sonication, resulting in penetration by solvent. However, the individual intrasheet hydrogen bonds are not so easily broken. The single cellobiose model gives a qualitatively different picture demonstrating that when attempting to understand the properties of cellulose nanofibrils, a realistic, extended structure is required - truncation in terms of either the number of elementary cellulose chains or their lengths can change the shape of the potential energy surfaces significantly.

5. CONCLUSIONS

We have presented a hybrid MPI-OpenMP implementation of the ONETEP software package and illustrated its performance on a range of high performance computing platforms. Our work has focused on the most computationally demanding routines within the code, namely FFT box operations, operations performed on sparse matrices, the calculation of integrals, and the Ewald summation. We have demonstrated the ability of the hybrid MPI-OpenMP code to scale to the regime of tens of thousands of atoms on tens of thousands of cores, a scale that is out of the reach of the MPI-only code. This is achieved by overcoming restrictions that arise from both software and hardware limitations.

The limitations of the MPI-only code arise from an algorithmic restriction to more than one atom per core and the technical issue of high communication costs arising from the use of thousands of MPI ranks. Within the hybrid MPI-OpenMP code the former restriction is lifted entirely and the latter restriction is severely reduced, as the possible number of cores per MPI process is increased from just one up to potentially the number of cores per node (or per NUMA region if applicable). Hardware restrictions such as a low quantity of RAM per core are overcome by utilization of shared memory by the hybrid MPI-OpenMP code. This is a very important issue as it means that machines such as BlueJoule may be used much more efficiently than previously possible.

The limitations in the current hybrid MPI-OpenMP implementation are 3-fold. The first issue, poor parallel efficiency at high MPI process counts, appears to arise from the scaling of the sparse matrix product code. Addressing this issue will further improve the scale of the calculations for which ONETEP may be used. Second, there is a parallel efficiency hit per added thread resulting from serialisation of MPI communications within threaded regions: this may be addressed using thread-aware calls to the MPI library. The final issue, the strong scaling at lower MPI process counts but a high number of cores per atom, is, in part, caused by the limitation of the FFT box code to a single level of parallelism (each FFT box operation is performed entirely by a single core): a second level of parallelism is technically possible and in future implementations will increase performance significantly; however, the limitation of a single MPI process per NUMA region would still apply.

Future developments to address these limitations include the application of OpenMP parallelism to other areas of the code, inclusion of a second level of OpenMP parallelism within the FFT box operations themselves, and the combination of the GPU accelerated version of the code described elsewhere[23] with these developments.

In summary, the development of a hybrid MPI-OpenMP implementation of the ONETEP code improves the applicability of ONETEP significantly, enabling shorter "time to solution" and permitting users to utilize powerful HPC resources more efficiently. This is exemplified by the calculations we performed on the 13,969 atom amyloid fibril protein on the BlueJoule BlueGene/Q machine which performed 6× faster with an 4× increase in resources due to the inability of the MPI-only implementation to fully utilize the cores on a node. These developments also facilitate the

application of the ONETEP code to much larger problem sizes than previously attempted, illustrated by calculations performed on systems containing 41,907 atoms.

We have applied the hybrid MPI-OpenMP code to simulate cellulose nanofibrils in order to investigate the mechanism by which sheets of elementary cellulose chains are separated when undergoing sonication. We have found that for this biological structure it is important to use large models, such as those used here with 21,276 atoms, in order to correctly describe the interactions that collectively determine the structural properties of the fibrils. It would have been nearly impossible to perform these calculations with the MPI-only implementation of ONETEP, but with the developments described here they are now feasible.

AUTHOR INFORMATION

Corresponding Author

*E-mail: c.skylaris@soton.ac.uk.

Notes

The authors declare no competing financial interest.

ACKNOWLEDGMENTS

We are grateful to iSolutions unit of the University of Southampton for access to the Iridis 4 supercomputer and administrative support. We acknowledge access to the ARCHER super computer via the UKCP consortium (EPSRC grant numbers EP/K013556/1 and EP/K014560/1) and to the BlueJoule super computer at the STFC Hartree Centre, Daresbury Laboratory, UK. The STFC Hartree Centre is a research collaboratory in association with IBM providing High Performance Computing platforms funded by the UK's investment in e-Infrastructure. K.W. would like to thank the Engineering and Physical Sciences Research Council for postdoctoral funding under grant number EP/I006613/1. NDMH acknowledges the support of the Winton Programme for the Physics of Sustainability.

REFERENCES

(1) Skylaris, C.-K.; Haynes, P. D.; Mostofi, A. A.; Payne, M. C. *J. Chem. Phys.* **2005**, *122*, 084119.
(2) Bowler, D. R.; Miyazaki, T.; Gillan, M. J. *J. Phys.: Condens. Matter* **2002**, *14*, 2781−2798.
(3) Soler, J. M.; Artacho, E.; Gale, J. D.; Garcia, A.; Junquera, J.; Ordejon, P.; Sanchez-Portal, D. *J. Phys.: Condens. Matter* **2002**, *14*, 2745−2779.
(4) Ozaki, T. *Phys. Rev. B: Condens. Matter* **2003**, *67*, 155108−155113.
(5) Rudberg, E.; Rubensson, E. H.; Sałek, P. *J. Chem. Theory Comput.* **2011**, *7*, 340−350.
(6) Hutter, J.; Iannuzzi, M.; Schiffmann, F.; VandeVondele, J. *Wiley Interdiscip. Rev.: Comput. Mol. Sci.* **2014**, *4*, 15−25.
(7) Genovese, L.; Neelov, A.; Goedecker, S.; Deutsch, T.; Ghasemi, S. A.; Willand, A.; Caliste, D.; Zilberberg, O.; Rayson, M.; Bergman, A.; Schneider, R. *J. Chem. Phys.* **2008**, *129*, 014109.
(8) Goedecker, S. *Rev. Mod. Phys.* **1999**, *71*, 1085−1123.
(9) Goedecker, S.; Scuseria, G. E. *Comput. Sci. Eng.* **2003**, *5*, 14−21.
(10) Galli, G.; Parrinello, M. *Phys. Rev. Lett.* **1992**, *69*, 3547−3550.
(11) Bowler, D. R.; Miyazaki, T. *Rep. Prog. Phys.* **2012**, *75*, 036503.
(12) Dagum, L.; Menon, R. *IEEE Comput. Sci. Eng.* **1998**, *5*, 46−55.
(13) *MPI Forum: A Message-Passing Interface Standard*; 1994.
(14) Fox, S. J.; Pittock, C.; Fox, T.; Tautermann, C. S.; Malcolm, N.; Skylaris, C.-K. *J. Chem. Phys.* **2011**, *135*, 224107.
(15) Hine, N. D. M.; Robinson, M.; Haynes, P. D.; Skylaris, C.-K.; Payne, M. C.; Mostofi, A. A. *Phys. Rev. B: Condens. Matter* **2011**, *83*, 195102.
(16) Ratcliff, L. E.; Hine, N. D. M.; Haynes, P. D. *Phys. Rev. B: Condens. Matter* **2011**, *84*, 165131.
(17) Kohn, W. *Phys. Rev. Lett.* **1996**, *76*, 3168−3171.
(18) Prodan, E.; Kohn, W. *Proc. Natl. Acad. Sci. U.S.A.* **2005**, *102*, 11635−11638.
(19) Skylaris, C.-K.; Mostofi, A. A.; Haynes, P. D.; Diéguez, O.; Payne, M. C. *Phys. Rev. B: Condens. Mater. Phys.* **2002**, *66*, 035119.
(20) Baye, D.; Heenen, P. H. *J. Phys. A, Math. Gen.* **1986**, *19*, 2041.
(21) Mostofi, A. A.; Haynes, P. D.; Skylaris, C.-K.; Payne, M. C. *J. Chem. Phys.* **2003**, *119*, 8842−8848.
(22) Skylaris, C.-K.; Mostofi, A. A.; Haynes, P. D.; Pickard, C. J.; Payne, M. C. *Comput. Phys. Commun.* **2001**, *140*, 315−322.
(23) Wilkinson, K.; Skylaris, C.-K. *J. Comput. Chem.* **2013**, *34*, 2446−2459.
(24) Ozaki, T. *Phys. Rev. B: Condens. Matter* **2001**, *64*, 195110.
(25) Palser, A. H. R.; Manolopoulos, D. E. *Phys. Rev. B: Condens. Matter* **1998**, *58*, 12704.
(26) Li, X.; Nunes, R.; Vanderbilt, D. *Phys. Rev. B: Condens. Matter* **1993**, *47*, 10891.
(27) Hine, N. D. M.; Haynes, P. D.; Mostofi, A. A.; Skylaris, C.-K.; Payne, M. C. *Comput. Phys. Commun.* **2009**, *180*, 1041−1053.
(28) Hine, N. D. M.; Haynes, P. D.; Mostofi, A. A.; Payne, M. C. *J. Chem. Phys.* **2010**, *133*, 114111.
(29) Ewald, P. P. *Ann. Phys. (Berlin, Ger.)* **1921**, *369*, 253−287.
(30) Skylaris, C.-K.; Haynes, P. D.; Mostofi, A. A.; Payne, M. C. *Phys. Status Solidi B* **2006**, *243*, 973−988.
(31) Amdahl, G. M. *Validity of the Single Processor Approach to Achieving Large Scale Computing Capabilities.* In Proceedings of the April 18−20, 1967, Spring Joint Computer Conference. New York, NY, USA, 1967; pp 483−485.
(32) Bowler, D. R.; Miyazaki, T. *J. Phys.: Condens. Matter* **2010**, *22*, 074207.
(33) Bock, N.; Challacombe, M.; Gan, C. K.; Henkelman, G.; Nemeth, K.; Niklasson, A. M. N.; Odell, A.; Schwegler, E.; Tymczak, C. J.; Weber, V. FREEON, Los Alamos National Laboratory (LA-CC 01-2; LA-CC-04-086), Copyright University of California, 2013. http://www.freeon.org/ (accessed Sept 14, 2014).
(34) Corsetti, F. *PLoS ONE* **2014**, *9*.
(35) VandeVondele, J.; Borstnik, U.; Hutter, J. *J. Chem. Theory Comput.* **2012**, *8*, 3565−3573.
(36) Borstnik, U.; VandeVondele, J.; Weber, V.; Hutter, J. *Parallel Computing* **2014**, *40*, 47−58.
(37) Farrell, A. E.; Plevin, R. J.; Turner, B. T.; Jones, A. D.; O'Hare, M.; Kammen, D. M. *Science* **2006**, *311*, 506−508.
(38) Huber, G. W.; Iborra, S.; Corma, A. *Chem. Rev.* **2006**, *106*, 4044−4098 PMID: 16967928..
(39) Domingues, R. M. A.; Gomes, M. E.; Reis, R. L. *Biomacromolecules* **2014**, *15*, 2327−2346.
(40) Ding, S.-Y.; Himmel, M. E. *J. Agric. Food Chem.* **2006**, *54*, 597−606 PMID: 16448156.
(41) Li, Q.; Renneckar, S. *Biomacromolecules* **2011**, *12*, 650−659.
(42) Li, Q.; Renneckar, S. *Cellulose* **2009**, *16*, 1025−1032.
(43) Qian, X.; Ding, S.-Y.; Nimlos, M. R.; Johnson, D. K.; Himmel, M. E. *Macromolecules* **2005**, *38*, 10580−10589.
(44) Devarajan, A.; Markutsya, S.; Lamm, M. H.; Cheng, X.; Smith, J. C.; Baluyut, J. Y.; Kholod, Y.; Gordon, M. S.; Windus, T. L. *J. Phys. Chem. B* **2013**, *117*, 10430−10443.
(45) Bledzki, A.; Gassan, J. *Prog. Polym. Sci.* **1999**, *24*, 221−274.
(46) Gomes, T. C. F.; Skaf, M. S. *J. Comput. Chem.* **2012**, *33*, 1338−1346.
(47) Grimme, S. *J. Comput. Chem.* **2006**, *27*, 1787−1799.