2010 12th IEEE International Conference on High Performance Computing and Communications

Parallel Computational Modelling of Inelastic Neutron Scattering in Multi-Node and Multi-Core Architectures

Michael T. Garba[*], Horacio González–Vélez[*] and Daniel L. Roach[†]
[*] School of Computing, Robert Gordon University, Aberdeen AB25 1HG, UK
[†] Physics and Materials Research Centre, University of Salford, Salford M5 4WT, UK
E-mail: {m.t.garba,h.gonzalez-velez}@rgu.ac.uk {d.roach}@salford.ac.uk

***Abstract*—This paper examines the initial parallel implementation of SCATTER, a computationally intensive inelastic neutron scattering routine with polycrystalline averaging capability, for the General Utility Lattice Program (GULP). Of particular importance to structural investigation on the atomic scale, this work identifies the computational features of SCATTER relevant to a parallel implementation and presents initial results from performance tests on multi-core and multi-node environments. Our initial approach exhibits near-linear scalability up to 256 MPI processes for a significant model.**

***Index Terms*—Computational Physics; Lattice Dynamics; Neutron Scattering; Parallel Programming; Scientific computing; Algorithmic Skeletons; Multi-Core Processors**

I. Introduction

An increasingly significant role in modern scientific investigation and engineering is played by computational simulation techniques. With a character distinct from the traditional approaches that are theory and experimentation, scientific computing and numerical modelling have emerged as established means of research in their own right within the physical sciences. Often capable of providing answers to analytically intractable problems unamenable to these traditional approaches, they render further insights into closed-form analytical solutions where they exist. A consequence of the trend towards more powerful machines and the evolution of these techniques is that predictions can often be made ahead of experimentation on the basis of computer models.

The General Utility Lattice Program (GULP) is a generalised symmetry-adapted lattice dynamics and simulation environment for the study of solid materials that provides a number of routines for the modelling, prediction and interpretation of experimental data in the study of atomic, molecular, and bulk crystalline structures [1]. As a research tool in the physical sciences with interdisciplinary applications in chemistry, physics, and material science, GULP is cited with increasing frequency in research within these domains. The regular introduction of new routines and enhancements to existing functionality is the result of an active development effort.

GULP is available as an open source software package for non-commercial use and as part of the Materials Studio from Accelrys. Input-file driven routines range from simulation to model fitting. Execution options, required and optional simulation parameters, general program options, and structural information are specified in the input deck. GULP outputs results to file or standard output alongside intermediate files that may contain data relevant to their interpretation. The time and space complexities of these GULP routines are directly correlated to the parametric and structural characteristics of the solid under investigation.

SCATTER, a new routine, makes extensive use of existing functionality to bring coherent and incoherent inelastic neutron scattering (INS) capabilities for lattice models to GULP [2], [3]. Until recently, INS was considered impractical on account of its significant computational requirements and the high cost and general unavailability of neutron scattering experimental facilities and instrumentation. INS modelling is experiencing increasing popularity for structural determination in solids among the materials science community, as a result of its suitability to problems such as those that occur in the study of nano-materials. A growing need has been created for tools that aid in the swift interpretation of complex data generated [2], nonetheless the implementation of efficient parallel INS solutions remains an open problem.

The contribution of this paper is the deployment of a performance-oriented parallel version of SCATTER for both multi-core and multi-node architectures. As parallel programming aims to capitalise on concurrency, the simultaneous execution of different components of SCATTER shall be mapped to distinct processing elements in order to improve overall performance.

This paper briefly outlines the significance of GULP as a tool in scientific computing and the particular contribution of SCATTER. It then examines the operation of SCATTER, presents an approach to its parallelisation and the initial results of execution and scalability testing in the SARA supercomputing facilities in the Netherlands.

II. Background

GULP is intended to solve a range of problems in molecular modelling and experimental data interpretation, with routines covering potential applications that range from simulation to model fitting. Symmetry is exploited within GULP to minimise redundant computation and provide a performance advantage over existing software in the same problem domain[1].

978-0-7695-4214-0/10 $26.00 © 2010 IEEE
DOI 10.1109/HPCC.2010.45

GULP includes complex MD capabilities to determine trajectories in a molecular ensemble. These routines compute positions and momenta of interacting particles over small discrete time increments until system-wide equilibrium is attained. This solution of the n-body problem is the only means of tracing the motion of a large number of interacting particles, and MD runs involving millions of atoms are not uncommon with applications spanning different scientific disciplines.

Heavily computationally intensive for large problem sizes, MD has yielded a number of approaches aimed at reducing its computational footprint to acceptable levels while maintaining its scalability. Ab-initio MD require solutions of the Schrödinger equation from quantum mechanics at each time-step, using the Born-Oppenheimer approximation to permit the calculation of a possible numerical solution [4]. Different authors have employed ab-initio approaches with GULP to produce several computational models [5], [6].

While ab-initio methods can be accurate and capable of modelling quantum phenomena such as molecular bonding, they disregard the contributions of longer range potentials to improve performance at the cost of accuracy and predictive power.

Moreover, INS is itself a powerful investigative tool that generates vast amounts of data that are a challenge to interpret. SCATTER models the interaction of neutrons incident on single crystal and polycrystalline samples in reciprocal space, achieving this by extensive use of existing GULP routines [2], [3]. SCATTER allows the comparison and refinement of theoretical models against experimentally obtained results on the accuracy of which space sampling resolution has direct bearing. However, determinations of the scattering function for all values of a large set of magnitudes and directions create a significant computational load, frequently requiring days to weeks of execution time. The analogously computationally demanding nature of MD [7] is believed to be of particular relevance to an implementation of SCATTER.

Other libraries for INS modelling include PHONON [8] and a-CLIMAX [9]. However, these packages focus on modelling INS datasets from *ab-initio* and density functional theory techniques, and lack semi-empirical modelling capabilities, which are of core interest in the study of a wide range of technological nanocarbons, hydrogen storage materials and carbon composite materials.

From an architectural point of view, the advent of multi-core processors, chip multiprocessors, and multi-node clusters and constellations has steeply increased the number of concurrent processors available to a single application. From a single node perspective, dozens of cores are becoming progressively more commonplace. Consequently, the development of effective simulation tools, which can be staged in a scalable, structured, fashion and are capable of exploiting parallelism on multiple architectural levels, has remained a target of sustained effort.

Arguably, a better compromise between accuracy and computational demand is obtained with the semi-empirical approach, as used in SCATTER, but scant research has been devoted to the use of these methods [2]. As the time and space complexities of semi-empirical MD methods remain an open problem in computational science, we would like to explore the application of structured parallelism to improve the SCATTER routines for the molecular simulation section of GULP in multi-core and multi-node architectures.

III. METHODOLOGY

SCATTER calculates the coherent and incoherent scattering intensities, denoted by expressions (1) and (2) respectively, for a momentum transfer vector Q representing the momentum change between incident and scattered wave vectors, and a vibrational frequency of the quantised lattice vibration (or phonon) created or annihilated by this scattering event. This frequency change is directly related to the modulus of the energy transfer between the target material and the scattered neutron, as determined by energy conservation and the principle of detailed balance [10], [3].

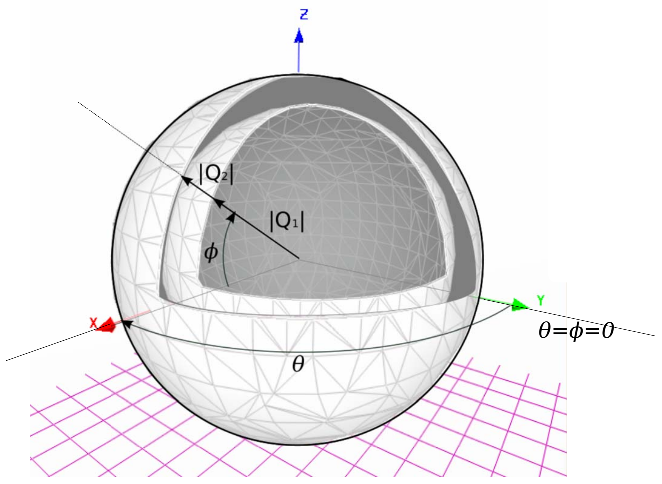

Figure 1. Reciprocal Space Onion Sampling in SCATTER. Concentric shells are traced in reciprocal space as data points are gathered for increasing magnitudes and orientations of the momentum transfer vector, Q, in spherical polar coordinates.

$$S_{coh}(Q,\omega) \tag{1}$$

$$S_{inc}(Q,\omega) \tag{2}$$

The coherent component of the scattering intensity in expression (1) takes into account cross-correlative pairwise interactions of the nuclei in the system and describes inelastic interference effects which provide information on the positions and vibrational modes of planes of atoms. Expression (2) represents the self-correlative, incoherent component of the scattering intensity, the vibrational contribution of individual atoms considered in isolation, without this interference term. SCATTER is capable of determining both cross-correlative and self-correlative components, however the primary application of this method is for coherent scattering computations and the majority of simulation runs currently involve this [2].

While it is possible to perform these calculations along fixed directions in reciprocal space, as is the case for a single crystal sample in an experimental configuration, it is usually of greater interest, and a better fit to experimental conditions where only polycrystalline samples may be available, to perform this determination over a range of values for magnitudes and spatial orientations of the momentum transfer vector Q in three dimensions.

One space sampling technique implemented in SCATTER is the Reciprocal Space Onion (RSO) sampling method illustrated in Figure 1. RSO provides data points for this computation, taking values of Q, as it is rotated about a series of concentric spheres by varying its magnitude $|Q|$ and angles θ and ϕ in a spherical polar coordinate system[3]. This traces concentric shells that bear close correspondence to the space sampling performed by the actual experimental triple-axis spectrometer around the target material. Our parallelisation effort will use the RSO sampling method.

In practice, better results are obtained by the choice of a greater number of sample points as, otherwise, artifacts may begin to appear in the final output for higher values of $|Q|$, where space is sampled in a sparser manner. Significant computational load results from the need to invoke several GULP routines on large dynamical matrices for each determination of expression (1) over the large range of values for Q generated by space sampling. A trade-off is necessitated between model resolution and execution time.

SCATTER concludes with a polycrystalline or powder averaging, allowing the generation of 3D plots comparable to empirically determined results. This final phase of execution involves the creation of a 3D histogram, matching values of expression (1) with corresponding values of $|Q|$ and ω with the use of separate visualisation tools. A key additional feature is the generation of computational log and data output files, (`sq-onions`), detailing the progression of intermediate computations at several stages. This data is capable of significantly enhancing an investigator's ability to arrive at deductions on the basis of the model.

A. *Identifying parallelism*

GULP implements a parallel MD routine based on MPI, originally documented to support up to 64 concurrent processes with rank-based task distribution. It evenly divides the workload among the available processes and furnished structured calls to the MPI communication primitives unify results at the conclusion of computation. Reasonable performance gains have been reported on a homogeneous multiprocessor system for the parallel MD routines in GULP [11], [12].

Potential parallelism is identifiable within SCATTER by an examination of the nature of dominant tasks in a typical polycrystalline run. Evaluations of the coherent and incoherent scattering magnitudes occur in loops over sampled space, with no data dependence between iterations, that indicate strong potential task parallelism. This leads naturally to a data decomposition scheme in which each task handles evaluations for only a limited subset of sampled space. This absence of data dependencies between evaluations makes SCATTER embarrassingly parallel. However, the I/O requirements of the massive output datasets generated place real constraints on achievable performance that is dictated by the underlying hardware.

It is considered highly desirable that a parallel SCATTER remains compatible on an architectural level with the existing parallelism within GULP to address maintainability issues and achieve a reduction of duplicated effort. However, it is not sine qua non. Furthermore, as it is now known that most assumptions that hold for multi-node environments are not entirely applicable to multi-core architectures [13], our approach must efficiently target both paradigms.

IV. IMPLEMENTATION

The parallel execution of SCATTER begins with the initial program input deck, of relatively small size, read via standard input and distributed to cooperating processes. The root process uses control of the actual standard input and output channels to perform this necessary initialisation before the start of computation. Figure 2 illustrates this distribution in an MPI broadcast operation. This approach to parallel input is already implemented within GULP, and it is crucial that the parallel SCATTER version is compliant with this interface.

With complete details of all execution parameters, each process independently generates the entire global sample space and proceeds to the actual SCATTER calculation. Task distribution is cyclical over concentric shells by rank. Fine-grain parallelism is impractical at present as each shell forms an integral input to subsequent invocations of intrinsic routines that operate on dynamical multi-dimensional arrays of significant size and computational requirements. This phase dominates the execution time of the SCATTER run.

As the intermediate results of these calculations are of analytical importance, MPI/IO provides scalable, distributed, simultaneous output of this large dataset.

In the final stages, each process generates a three-dimensional $|Q|$, ω "histogram" representing the polycrystalline average for the relevant region of the partitioned sampled space that represents a local contribution to the ultimate result. A global reduction operation communicates these results to the root process in a final summation that merges the data generated from each task into a polycrystalline average.

Our implementation of the parallel SCATTER routine has followed the SPMD model. Using the existing optional MPI bindings in GULP, it exploits the absence of data dependencies between successive evaluations of expression (1) in a replicated data pattern with reduction via `MPI_REDUCE`.

V. EVALUATION

We created the first development version of SCATTER with MPI support in early 2010. We initially tested the pure multi-node feasibility on a iMac cluster, forcing a one core per node mapping, and the multi-core behaviour on a IBM BladeCenter. Both versions were compiled with gfortran only.

Table I
MULTI-NODE AND MULTI-CORE TEST MACHINE SPECIFICATIONS

	iMac Cluster	IBM BladeCenter JS21	IBM pSeries 575 (Huygens)
Nodes	8	1	101
Processing Elements per Node	2	4	32
Processor Speed	2.66 GHz	2.3 GHz	4.7 GHz
Processor Architecture	Intel Core 2 Duo	IBM PowerPC 970MP dual-core	IBM Power6
RAM	2 GB	16 GB	128 GB
Network	100baseTX	N/A	Infiniband 160 Gbit/sec
Operating System	Mac OS X 10.5.8	Redhat Linux (kernel 26.18-8)	GNU Linux Kernel 2.6.27
Compiler	GCC gfortran 4.4.1	GCC gfortran 4.1.2	GCC gfortran 4.3.2 (-O3) IBM XL Fortran for Linux, V12.1 (-O3 -qstrict -qarch=auto -qtune=auto)
MPI Version	OpenMPI 1.4.2	OpenMPI 1.4.2	OpenMPI 1.3.3

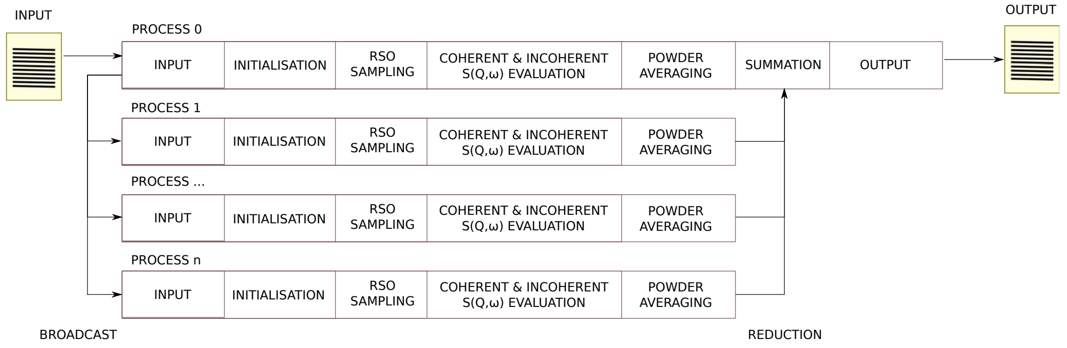

Figure 2. Communication and communication structure between MPI processes in SCATTER.

Their software and hardware specifications are described in Table I.

Subsequently, the complete performance evaluation has been conducted on the PRACE supercomputing prototype (`huygens`) located at SARA, the Dutch National High Performance Computing and e-Science Support Centre. This prototype has large shared memory (4-8 GB/core) and fast I/O configuration with the new IBM Power6 processors and IBM Power Cluster fat node architecture. In order to find the most suitable system optimisations, both the GNU Fortran and the IBM XL Fortran compiler have been employed as shown in the last column of Table I.

As described in its webpage [14], the `huygens` system has 1664 dual core processors, equivalent to 3328 cores, 15.25 terabytes of main memory, and 700 terabytes of disk space. Being a multi-node environment with multiple cores, it represents parallelism on multiple levels.

The input dataset was a 40-atom 10,10 Carbon Nanotube model with $\left(\frac{|Q|_{max}-|Q|_{min}}{\delta|Q|}\right)$= 256 shells and $\frac{2\pi}{\delta\theta}$=200 angular steps, as originally illustrated in Figure 1. An initial calibration run with coarse angular resolution provided an estimated time-to-completion for the full model using equation (3).

$$t \approx k\left(\frac{|Q|_{max}-|Q|_{min}}{\delta|Q|}\right)\times\left(\frac{2\pi}{\delta\theta}\right)^2 \qquad (3)$$

In equation 3, t is the approximate completion time, k is a constant for a given execution environment and model, $|Q|_{max}-|Q|_{min}$ is the difference between the maximum and minimum values of the momentum transfer vector, δQ is the finite increment in momentum transfer between successive RSO shells and $\delta\theta = \delta\phi$ is the finite change in angles of the momentum transfer vector.

It is also important to note that the SCATTER I/O requirements are primarily defined by the same relationship expressed in equation 3. This dataset is composed by the intermediate calculation logs (`sq-onions`) and the final summary file, a relatively small file in comparison.

A. Discussion

The results of the initial calibration run, available in Figure 3, indicate significantly improved performance with the IBM XL Fortran compiler over GNU Fortran. This difference can be attributed to the ability of the IBM XL Fortran compiler to exploit the on-chip parallelism and other architecture-specific optimisations.

The estimated completion time for the full model was originally 2733 hours (114 days) with the GNU Fortran version

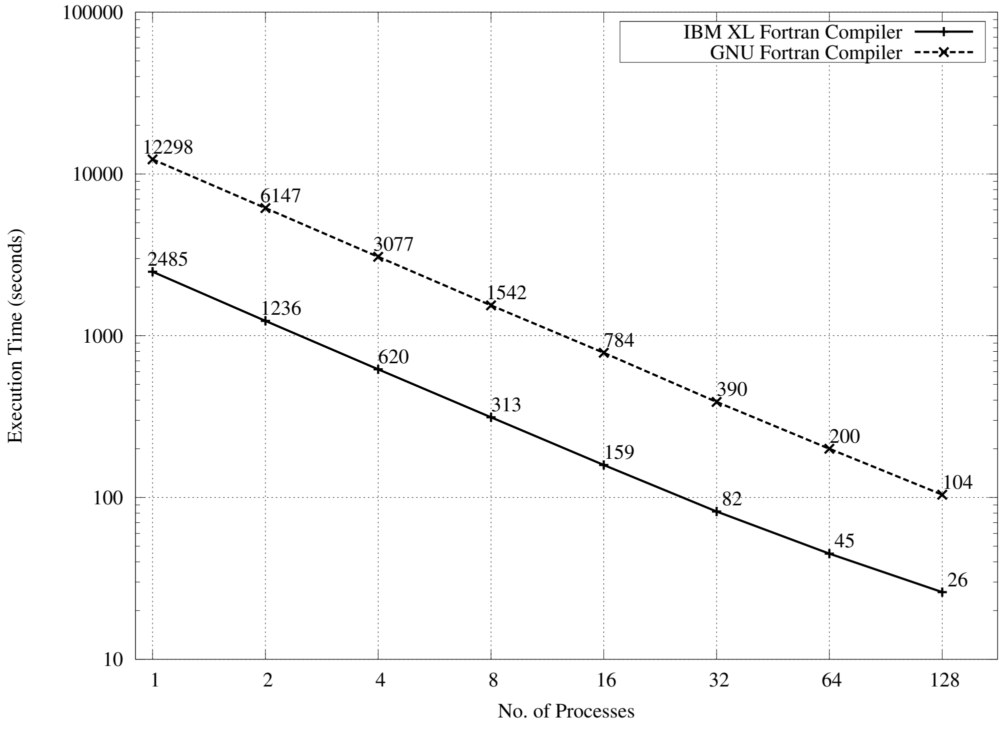

Figure 3. Performance of GNU Fortran and IBM XL Fortran (with architecture-specific optimisations) versions of GULP with 2^n processes a (10,10) Carbon Nanotube model in a coarse reciprocal space onion sampling for runtime estimation. $\left(\frac{|Q|_{max}-|Q|_{min}}{\delta|Q|}\right)$= 128 and $\frac{2\pi}{\delta\theta}$=10. Cell parallelism achieves significant speed-up for SCATTER kernel evaluation

and 552 hours (23 days) with the IBM XL Fortran version on a single processor. This calibration also revealed near-linear scaling up to 128 processes with a small discontinuity between 32 and 64 processes, indicating moderately degraded performance at the intra-node to inter-node transition boundary.

Execution times for the actual model on $32, 64, 128$ and 256 processes are presented in Figure 4, generating a 120GB output dataset in each run and completing in 7405 seconds—just over 2 hours—for the 256-process case. This version, compiled with the IBM XL Fortran compiler, exhibits linear scaling in the multi-node case, despite the fact that it is memory and I/O intensive. We believe that the fast interconnection at Huygens and the MPI implementation have made a significant difference in easing any potential bottlenecks.

VI. FINAL REMARKS

Future work intends to explore the two main directions of parallel construct refinement and scalability.

In terms of parallel constructs, further development is planned to take advantage of the intrinsic SCATTER parallelism. We intend to deploy algorithmic skeletons [15], possibly in the form of a refined task farm [16], with the objective of achieving closer-to-optimal resource utilisation. Subsequent versions will seek to examine the possibility of adaptive scheduling based on the structural and parametric characteristics of the physical models and execution environment. As successive evaluations are totally independent from each other, they can be regarded as a divisible load [17], we intend to explore their scheduling using known scheduling heuristics [18].

In terms of scalability, an evaluation of the performance of SCATTER and GULP for complex physical models, such as nano-structures and carbon compounds, is a natural progression. To the extent of our knowledge, the original GULP versions have been evaluated up to 64 processes, a bar that has been raised to 256 processes in SCATTER. Ideally, we would like to evaluate the performance of GULP+SCATTER with thousands of processes, running on as many nodes, in order to significantly improve the experimental resolution and range of potential applications. Achieving finer-grained parallelism to permit this will arguably require parallelising the large dynamical Hessian matrix calculations central to scattering kernel evaluation.

In summary, we have demonstrated that our deployment of a parallel SCATTER routine within GULP is theoretically feasible and practically necessary. The implementation under development has shifted the balance away from lengthy execution times and towards higher model resolutions, allowing material scientists to examine compounds of greater complexity and interest. Our preliminary results have indeed met the expectations that the current approach is an effective means of obtaining near-optimal performance in multi-core/multi-node architectures.

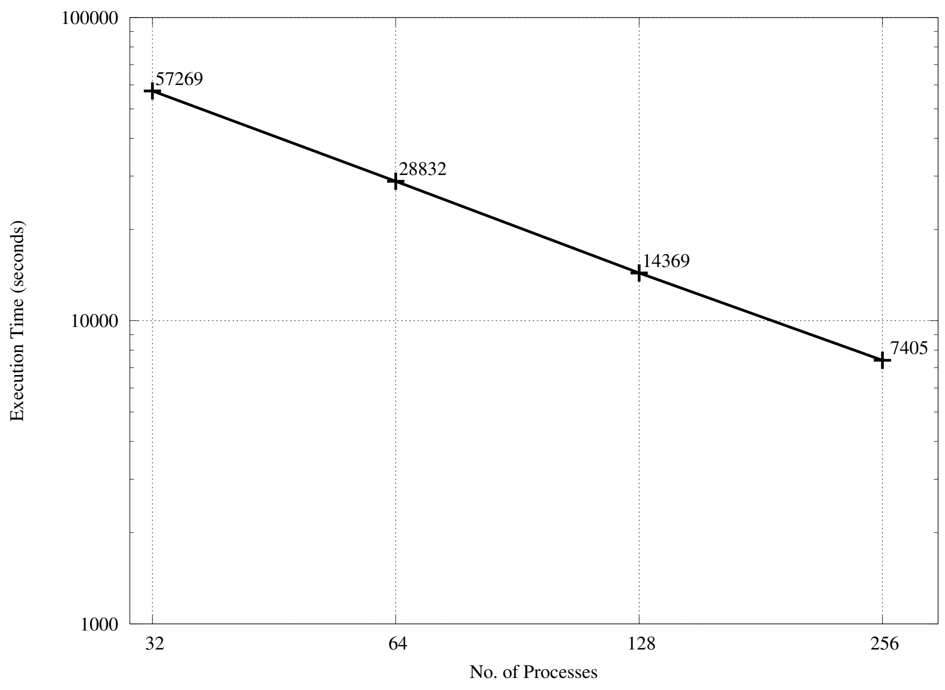

Figure 4. Computation times for the actual (10,10) Carbon Nanotube model with $\left(\frac{|Q|_{max}-|Q|_{min}}{\delta|Q|}\right)$= 256 and $\frac{2\pi}{\delta\theta}$=200, generating a 120GB dataset. Results indicate near-linear scalability for 32, 64, 128 and 256 processes in the multi-node configuration.

ACKNOWLEDGEMENTS

The authors would like to thank the Partnership for Advanced Computing in Europe (PRACE) for their support and grant of computing time in the SARA supercomputing facilities. The PRACE project receives funding from the EU's Seventh Framework Programme (FP7/2007-2013) under grant agreement no. RI-211528. The Authors would also like to acknowledge the support of a collaboration travel grant awarded by the STFC Collaborative Computational Project 5 (CCP5). One of the authors (DLR) would like to acknowledge the support of EPSRC (EP/G049130) in the development of the SCATTER code.

REFERENCES

[1] J. Gale, "GULP: A computer program for the symmetry-adapted simulation of solids," *Journal of the Chemical Society, Faraday Transactions*, vol. 93, no. 4, pp. 629–637, 1997.

[2] D. L. Roach, "Computational investigations of polycrystalline systems using inelastic neutron scattering techniques," Ph.D. dissertation, University of Salford, Salford M5 4WT, UK, 2006.

[3] D. L. Roach, J. Gale, and D. Ross, "Scatter: A New Inelastic Neutron Scattering Simulation Subroutine for GULP," *Neutron News*, vol. 18, no. 3, pp. 21–23, 2007.

[4] M. Griebel, S. Knapek, and G. Zumbusch, *Numerical simulation in molecular dynamics: Numerics, Algorithms, Parallelization, Applications*, ser. Texts in Computational Science and Engineering. Heidelberg: Springer-Verlag, 2007, vol. 5.

[5] B.-L. Huang and M. Kaviany, "Ab initio and molecular dynamics predictions for electron and phonon transport in bismuth telluride," *Phys. Rev. B*, vol. 77, no. 12, pp. 125–209, Mar 2008.

[6] F. Torres, P. Ugliengo, B. Civalleri, A. Terentyev, and C. Pisani, "A review of the computational studies of proton- and metal-exchanged chabazites as media for molecular hydrogen storage performed with the CRYSTAL code," *International Journal of Hydrogen Energy*, vol. 33, no. 2, pp. 746–754, 2008.

[7] D. C. Rapaport, *The art of molecular dynamics simulation*, 2nd ed. Cambridge: Cambridge University Press, 2004.

[8] K. Parlinski, Z. Q. Li, and Y. Kawazoe, "First-principles determination of the soft mode in cubic $zro2$," *Phys. Rev. Lett.*, vol. 78, no. 21, pp. 4063–4066, May 1997.

[9] D. Champion, J. Tomkinson, and G. Kearley, "a-CLIMAX: a new INS analysis tool," *Applied Physics A: Materials Science & Processing*, vol. 74, pp. 1302–1304, 2002.

[10] G. Squires, *Introduction to the theory of thermal neutron scattering*. Cambridge Univ. Press, 1978.

[11] D. Marx and J. Hutter, *Ab initio molecular dynamics: basic theory and advanced methods*. Cambridge Univ Pr, 2009.

[12] S. Plimpton and B. Hendrickson, "A new parallel method for molecular dynamics simulation of macromolecular systems," *Journal of Computational Chemistry*, vol. 17, no. 3, pp. 326–337, 1996.

[13] M. D. Hill, "Amdahl's law in the multicore era," in *HPCA-14 2008*. Salt Lake City: IEEE Computer Society, Feb. 2008, p. 187.

[14] SARA, "Description of the huygens system," Dutch National High Performance Computing and e-Science Support Center, https://subtrac.sara.nl/userdoc/wiki/huygens/description/, Web page, 2010, (Last Accessed: 30 Jun 2010).

[15] M. Cole, *Algorithmic Skeletons: Structured Management of Parallel Computation*, ser. Research Monographs in Parallel and Distributed Computing. London: MIT Press/Pitman, 1989.

[16] H. González-Vélez, "Self-adaptive skeletal task farm for computational grids," *Parallel Computing*, vol. 32, no. 7-8, pp. 479–490, 2006.

[17] V. Bharadwaj, D. Ghose, and T. G. Robertazzi, "Divisible load theory: A new paradigm for load scheduling in distributed systems," *Cluster Computing*, vol. 6, no. 1, pp. 7–17, 2003.

[18] H. González-Vélez and M. Cole, "Adaptive statistical scheduling of divisible workloads in heterogeneous systems," *Journal of Scheduling*, Oct. 2009, in Press. DOI: 10.1007/s10951-009-0138-4.