SACJ 29(3) December 2017
Research Article

Scalability of DL_POLY on High Performance Computing Platform

Mabule Samuel Mabakane[a, b], Daniel Mojalefa Moeketsi[a], Anton S. Lopis[b]

[a] Council for Scientific and Industrial Research, Centre for High Performance Computing, Cape Town, South Africa
[b] University of Cape Town, South Africa

ABSTRACT
This paper presents a case study on the scalability of several versions of the molecular dynamics code (DL_POLY) performed on South Africa's Centre for High Performance Computing e1350 IBM Linux cluster, Sun system and Lengau supercomputers. Within this study different problem sizes were designed and the same chosen systems were employed in order to test the performance of DL_POLY using weak and strong scalability. It was found that the speed-up results for the small systems were better than large systems on both Ethernet and Infiniband network. However, simulations of large systems in DL_POLY performed well using Infiniband network on Lengau cluster as compared to e1350 and Sun supercomputer.

Keywords: E1350 IBM Linux cluster, Sun, Lengau, Infiniband, DL_POLY, weak scalability, strong scalability

Categories: • *Computer systems organization ~ Parallel architectures* • *Computer systems organization ~ Multicore architectures*

Email:
Mabule Samuel Mabakane smabakane@csir.co.za (CORRESPONDING),
Daniel Mojalefa Moeketsi dmoeketsi@csir.co.za,
Anton S. Lopis alopis@csir.co.za

Article history:
Received: 21 Jul 2016
Accepted: 11 Oct 2017
Available online: 8 Dec 2017

1 INTRODUCTION

Over the past decades, understanding the performance of scientific codes (e.g. DL_POLY[1]) on distributed and shared memory computing platform has been a topic of research interest in the High Performance Computing (HPC) space (Hey, 1990; Heinrich et al., 1994; Jiang, Shan, & Pal Singh, 1997; Lange et al., 2011). Traditionally, it is expected that codes should scale linearly when one increases computational resources such as compute nodes or servers (Chamberlain, Chace, & Patil, 1998; Gropp & Snir, 2009). However, several studies showed different results compared to the anticipated linear scaling of codes (Agarwal et al., 1995; Kepner & Ahalt, 2004; Aldrich, Fernández-Villaverde, Ronald Gallant, & Rubio-Ramírez, 2011).

The actual performance of the computing system may also have an impact in the overall scaling of the scientific model. The performance of HPC systems was recognised in the early 1960s, when

Mabakane, M.S., Moeketsi, D.M. and Lopis, A.S. (2017). Scalability of DL_POLY on High Performance Computing Platform. *South African Computer Journal 29*(3), 81–94. https://doi.org/10.18489/sacj.v29i3.405

SACJ is a publication of the South African Institute of Computer Scientists and Information Technologists. ISSN 1015-7999 (print) ISSN 2313-7835 (online).
[1]For more information visit: https://www.scd.stfc.ac.uk/Pages/DL_POLY.aspx.

Gordon Moore (one of the founders of Intel) predicted that the performance of supercomputers would double every two years (J. Dongarra, Luszczek, & Petitet, 2003; J Dongarra, 2004; Kindratenko & Trancoso, 2011). In 1970s, the vector computer system was then introduced into the field of the supercomputer. In the late 1980s, different institutions started to show interest in parallel computing systems using distributed memory systems.

Beginning in the 1990s, multiprocessor systems joined the market and claimed to be better than vector systems (Hey, 1990; Oyanagi, 2002; Strohmaier, Dongarra, Meuer, & Simon, 2005). To provide more accurate information about the performance of HPC systems, the TOP500 list[2] was launched in 1993 to address the issues of fastest supercomputers in the world. Most recently, the list of supercomputers is published twice every year to determine the 500 most powerful computers in the world (J Dongarra, 2004; Oyanagi, 2002; Strohmaier et al., 2005; Kindratenko & Trancoso, 2011; Bertsimas, King, & Mazumder, 2016).

The first distributed memory system in South Africa (the e1350 IBM Linux cluster) was commissioned and put into service in mid 2007 to enable scientific users in South Africa to perform fast calculations within a short period of time. In 2009, it was replaced by another supercomputing system, namely the Sun cluster[3], which was then superseded by the Lengau Petaflop system currently also hosted at the Centre for High Performance Computing (CHPC).

The e1350 IBM Linux system consisted of 160 compute nodes connected to shared Storage Area Networks (SAN) of 94 Terabytes. Each compute node contained four Opteron 2.6 GHz processors and 16GB of memory. The peak performance of the IBM cluster was approximately 2.5 Teraflops/s and shared across the entire supercomputer using Infiniband (10 GB/s) and Ethernet (1 GB/s) networks. Networks may perform differently depending on the speed and technology within the network switches.

On the other hand, the Sun cluster consisted of different architectures, namely, Nehalem, Harpertown, Westmere and Dell. In this study, we focus on Nehalem and Harpertown, because these subsystems are the original architectures of the Sun supercomputer. For Nehalem nodes, each compute node was equipped with eight Intel Xeon processors (2.93 GHz) attached to 24GB of memory. The Harpertown nodes contained eight Intel Xeon processors (3.0 GHz) which were connected to 16GB of memory.

The performance (speed) of the Sun system was 61.5 Teraflops/s, which was shared through the Infiniband Network (40 GB/s) connected to the Storage Area Network of 400 Terabytes. However, CHPC's current Lengau cluster[4] performed up to 1.029 Petaflops/s and was equipped with Dell (Intel Xeon (R) E5-2690 V3 processors) and FAT nodes (Intel Xeon (R) E7-4850) processors. Each Dell node had 24 processors connected to 128 GB of memory and FAT nodes consists of 1 Terabyte of memory. Both e1350 and Sun clusters were distributed memory systems programmed with Message Passing Interface (MPI) used to spread tasks across the system. Lengau cluster is also a distributed memory system which utilises MPI for distribution of computational tasks.

[2]For more information about the list, please refer to: http://top500.org. More info about the list can be found on: http://www.isc-hpc.com.

[3]https://www.chpc.ac.za/index.php/resources/tsessebe-cluster

[4]https://www.chpc.ac.za/index.php/resources/lengau-cluster

https://doi.org/10.18489/sacj.v29i3.405

For the purpose of this study, we investigate the scaling of DL_POLY parallelised using MPI on e1350 IBM, Sun and Lengau clusters. DL_POLY uses MPI to create parallel processes and exchange data between the compute nodes of the e1350, Sun and Lengau cluster.

DL_POLY is a molecular dynamics (MD) package mainly used in fields such as chemistry and materials science and was developed at Daresbury Laboratory, United Kingdom (UK) under the auspices of the Council for the Central Laboratory of the Research Councils (W. Smith & Forester, 1996; W. Smith & Todorov, 2006). This MD application is used to simulate boxes of different types of atomic systems according to Newton's Laws of Motion.

The objective is to gain understanding into the performance of DL_POLY 2.18 and 3.09 codes on the e1350 IBM cluster when using the Infiniband and Ethernet networks respectively. Moreover, we also intend to understand how different versions of DL_POLY (namely 2.18, 3.09, 4.07 and Classic_1.9) perform on the e1350, Sun and Lengau clusters.

In the e1350, DL_POLY was compiled using PathScale Compiler[5] and parallelised using Open MPI[6]. On Sun cluster, DL_POLY was compiled using Intel Compiler[7] version 13.1 with Open MPI version 1.8.8, of which, it was further compiled using Intel Compiler version 16.0.1 with Open MPI version 1.10.2 on Lengau cluster.

2 DESIGN OF COMPUTATIONAL EXPERIMENT

All DL_POLY versions are coded in FORTRAN 90 with a modular programming approach and employ MPI parallelisation. Development of DL_POLY_2 has ended with the final version being DL_POLY_Classic 1.9. DL_POLY_2 (including versions 2.18 and Classic_1.9) employs Replicated Data parallelism without parallel I/O, and is most suited for systems comprising up to 30 000 atoms running on up to 100 processors (cores) (W. Smith & Todorov, 2006). DL_POLY_3 (including 3.09) utilises a static/equi-spacial Domain Decomposition parallelisation strategy in which the simulation cell (comprising the atoms, ions or molecules) is divided into quasi independent domains - this is achieved using a link cell approach employing "halo data" one link cell deep (Todorov, Smith, Trachenko, & Dove, 2006).

DL_POLY_3 is best suited for simulations of 100 000 to 1 000 000 atoms on up to 1000 processors (cores). Although DL_POLY 3.09 does use some parallel I/O via MPI file writing, subsequent versions (such as 3.10) have significantly better parallel I/O. DL_POLY_4 (such as version 4.07) is the direct successor of DL_POLY_3 which also includes concepts and functionality from DL_POLY_2. DL_POLY 4.07 hence employs the Domain Decomposition strategy, while I/O is fully parallel with netCDF being an option.

An important factor in determining the scalability of a code such as DL_POLY is the size of the system that is studied, viz., number of atoms simulated on a computational arena. To study the behavior of this atomic model, the method of executing scientific code using strong and weak

[5]https://github.com/pathscale
[6]https://www.open-mpi.org
[7]https://software.intel.com/en-us/intel-parallel-studio-xe

https://doi.org/10.18489/sacj.v29i3.405

scalability was invoked to study the effects of the network on the performance of DL_POLY on e1350 IBM Linux cluster (Mabakane, 2011).

Strong scalability involves doubling the number of nodes but maintaining the constant configuration size for the chosen scientific code, while weak scalability occurs when the number of nodes is doubled and the problem size is also concurrently increased (Bosilca, Delmas, Dongarra, & Langou, 2009; Varma, Wang, Mueller, Engelmann, & Scott, i2006). Using DL_POLY_2.18, the strong scalability of the model was studied in two different ways, namely, by using a small and large system (Tang, 2007).

The small configuration (8640 atoms) and a large problem sizes (69120 atoms) were selected to perform scalability tests because many CHPC DL_POLY users were utilising these configuration sizes to run simulations for different scientific purposes. In the small system, DL_POLY_2 was employed for a simulation comprising 8640 atoms (that is Sodium (Na) = 960, Potassium (K) = 960, Silicon (Si) = 1920, Oxygen (O) = 4800) (Bosilca et al., 2009). The chemical system of atoms were simulated in a cubic box of size 48.358 angstroms for each of its X, Y and Z axes. The temperature in all DL_POLY simulations was set to 1000K.

In the large system of DL_POLY_2, the cubic box was then increased in size from 48.358 cubic angstroms (small system) to 96.717 cubic angstroms (large system) in order to accommodate the higher number of atoms to be simulated. The number of atoms in this large system is 69120 atoms. This large atomic system consists of 7680 Na, 7680 K, 15360 Si and 38400 O atoms for a total of 69120 atoms (Todorov & Smith, 2004).

The scaling method of increasing the number of processors concurrently with problem size (weak scalability) has also been utilised to test the performance of DL_POLY_2.18. Seven different configurations were designed for this purpose as highlighted below in Table 1.

Table 1: Atomic compositions for testing weak scalability of DL_POLY versions 2.18 and 3.09

Atom	Weak_1	Weak_2	Weak_3	Weak_4	Weak_5	Weak_6	Weak_7
Na	212	424	848	1696	3392	6784	13568
K	212	424	848	1696	3392	6784	13568
Si	424	848	1696	3392	6784	13568	27136
O	1060	2120	4240	8480	16960	33920	67840
Total	1908	3816	7632	15264	30528	61056	122112

Starting from the first composition of disilicate glass (Weak_1), the number of atoms in the cell was doubled while the box size was appropriately increased in order to maintain the same density of atoms. This process of doubling the number of atoms was repeated until one reached the final disilicate composition (Weak_7). All seven atomic molecular dynamics simulations were performed using both Infiniband and Ethernet networks of the e1350 IBM Linux cluster. The same disilicate glass composition for DL_POLY_2.18 (small system) was employed for use in DL_POLY_3.09 simulation (small model) while the same system of higher composition used in DL_POLY_2.18 (large system) was also utilised in DL_POLY_3.09 (large system).

https://doi.org/10.18489/sacj.v29i3.405

Furthermore, the weak scalability of DL_POLY_3.09 has been tested on the e1350 IBM Linux cluster using the same setup of seven disilicate compositions, namely, Weak_1, Weak_2, Weak_3, Weak_4, Weak_5, Weak_6 and Weak_7, as shown in Table 1. These seven different compositions (configurations) have the corresponding number of atoms and cell sizes as used for DL_POLY_2.18 tests (weak scalability). On e1350, each atomic system was run 3 times in order to get a comfortable total execution time of the simulations. For Sun and Lengau, some of the runs were also repeated.

3 RESULTS AND DISCUSSIONS

As discussed, this MD code has been utilised to understand the effects of network when utilising the same disilicate compositions (small and large systems) and compositions of different sizes (weak scalability). The scaling of this MD code has been calculated utilising the following formula:

$$S(P) = \frac{T(1)}{T(P)} \tag{1}$$

The speed-up on P processors, S(P), is the ratio of the execution time on 1 processor, T(1), to the execution time on P processors, T(P) (Chamberlain et al., 1998). In an ideal situation, we expect the execution time of the application to scale linearly with the number of processors. As for DL_POLY_2.18 (small system), Figure 1 shows the impact of network on the speed of the MD simulations running on the e1350 cluster.

The scaling results of this with respect to both Infiniband and Ethernet networks indicate that the simulation speed-up is almost linear from 8 to 32 processors and thereafter deviates significantly below the ideal value from 32 to 256 processors. For the Infiniband network, the application's performance improves slightly from around 16 to 32 processors, before starting to decrease when using up until 256 processors. These DL_POLY_2.18 simulations of the small disilicate glass system show that performance is much better when using a small number of processors (at least 32 processors) than when using many processors (256 processors). This may be attributed to factors such as high load of communication between the nodes and design of the supercomputer used to simulate the model.

Figure 2 shows the complicated scaling results of DL_POLY_2.18 (large simulation) for both Infiniband and Ethernet networks of the e1350 cluster. In particular, from 4 to 256 processors we see values which exceed the expected speed-up value for both Infiniband and Ethernet network sharing communication choices within the e1350 IBM Linux cluster. These performance results show that the problem size exceeded the capabilities of the selected node used to perform the MD simulations (DL_POLY_2.18 large system), whereas employing additional nodes would ensure sufficient capacity.

DL_POLY_2.18 is designed to perform best up to 30 000 atoms, of which, the simulated problem size comprises of 69120 atoms. It is noticeable that performance of simulations of this large composition (69 120 atoms) got significantly worse from 4 to 256 processors, and then suddenly the performance decreased to almost 0 scaling when using 512 processors.

The results (Figure 2) indicate that DL_POLY_2.18 is not well suited for simulating relatively large problem sizes such as 69 120 atoms on the e1350 IBM Linux cluster. Hence, different configuration

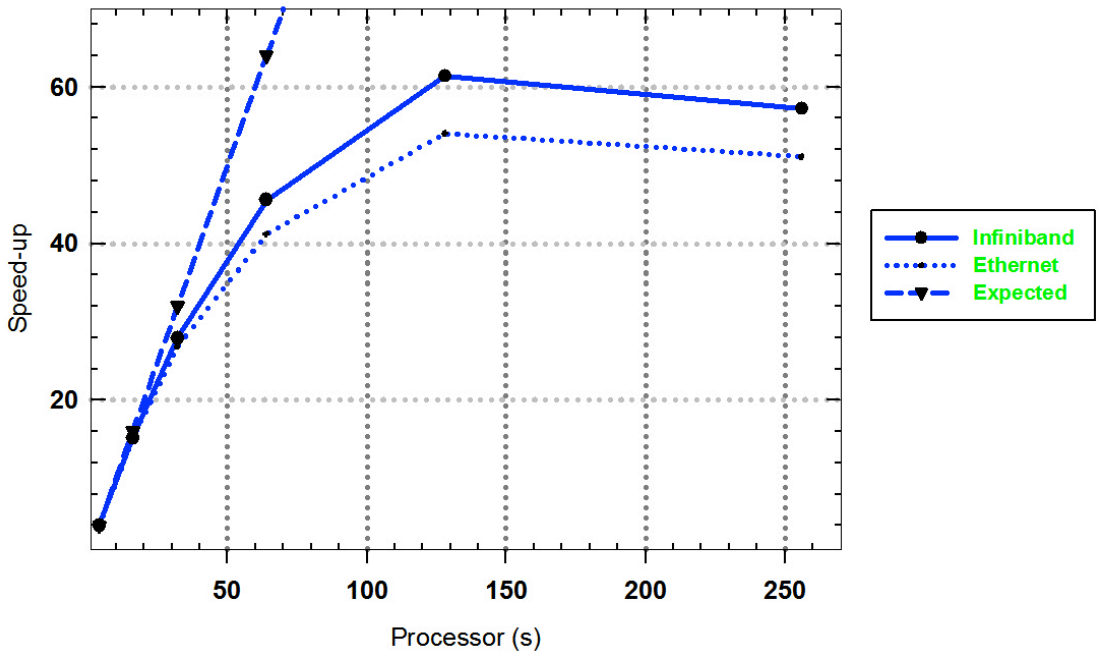

Figure 1: Strong scalability of DL_POLY_2.18 (small system)

sizes were designed and tested on different numbers of nodes in order to determine the optimal number of atoms that could provide good performance of the DL_POLY_2.18 model.

Figure 3 illustrates some performance results of different problem sizes simulated over Infiniband and Ethernet networks on the e1350 IBM Linux cluster. DL_POLY_2.18 simulations were performed for different disilicate compositions from Weak_1 up to Weak_6. Simulations for Weak_7 failed to run on the e1350 IBM Linux cluster, apparently owing to the system containing too many atoms (122112) to be handled by this version of molecular dynamics package on this cluster. It shows lower values than ideal scaling from 8 to 256 processors for both networks. The performance scaling is extremely similar for both networks, except that Ethernet network scales almost ideally when running on 16 and 32 processors. The reason for this poor scaling could be that the communication between the parallel processes was poorly coordinated within the system (W. Smith, Forester, & Todorov, 2008). Different codes are likely to perform differently especially when the codes do not have same features, for example in this case, DL_POLY_3.09 was tested after DL_POLY_2.18 in order to compare such differences.

Figure 4 shows that the results of the DL_POLY_3.19 small system (8640 atoms) for both networks indicates that the speed of the MD calculations deviate significantly from perfect scaling from around 16 to 64 processors. This result shows a huge difference in performance between DL_POLY_2.18 small system and DL_POLY_3.09 small simulation. From 4 to 64 processors, DL_POLY_3.09 (small system)

https://doi.org/10.18489/sacj.v29i3.405

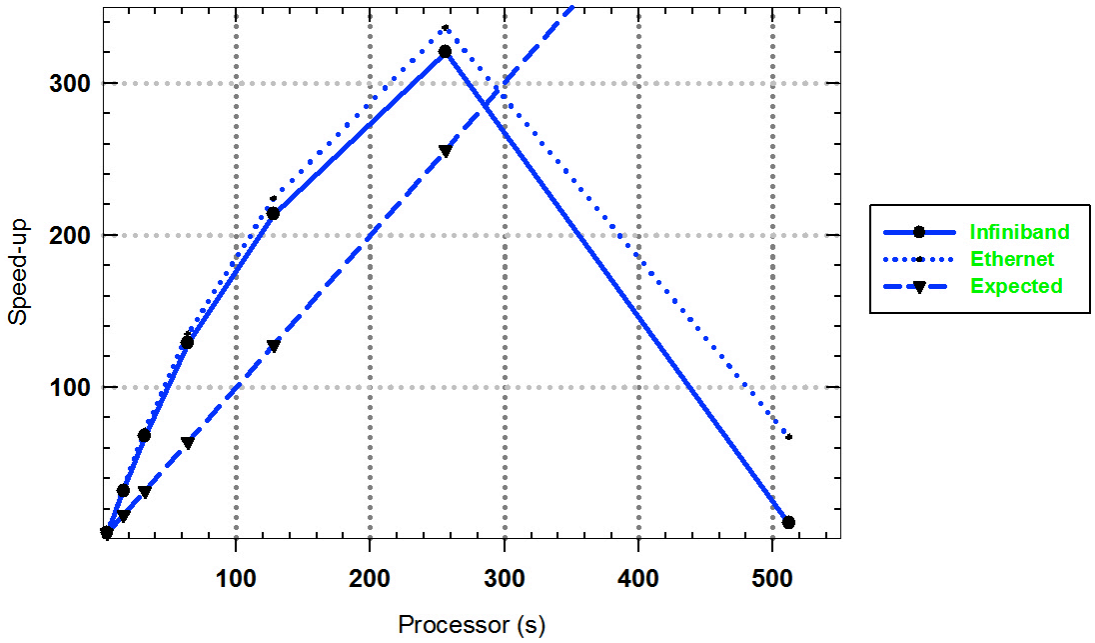

Figure 2: Strong scalability of DL_POLY_2.18 (large system)

managed to obtain a speed-up of less than 15 times on both Infiniband and Ethernet networks. The DL_POLY_2.18 small system obtained a far better speed-up of about 100 on 256 processors using both Infiniband and Ethernet network. Here the number of atoms in the system and number of processors used, are likely not too far from the range where DL_POLY_2.18 code is known to perform best (namely, 30 000 atoms and 100 processors). The small system of DL_POLY_3.09 was therefore modified by increasing the size of the starting configuration from 8640 to 69120 atoms (large system) in order to better determine the performance differences.

Figure 5 illustrates the performance results of the DL_POLY_3.09 simulation of the large system running on two different networks (Infiniband and Ethernet) on the e1350 IBM cluster. For both Infiniband and Ethernet networks, the results indicate that the simulation speed-up results are lower than the ideal value for 4 to 512 processors, but continue to increase with an increase in the number of nodes. This result tends to indicate that the chosen large configuration size could yield better scaling behavior when increasing the processing capabilities. However, this may also depend on the architecture used. As discussed in Section 2, different configuration sizes were designed using different numbers of atoms, and these were executed by using different processing capabilities, starting from 4 to 256 processors within the e1350 IBM Linux cluster.

Figure 6 illustrates the scaling results of different disilicate glass compositions simulated on Infiniband and Ethernet networks. The results indicate that the speed-up of the simulations yields dramatically lower values from 4 to 512 processors. In particular, scaling results (Figure 6) in-

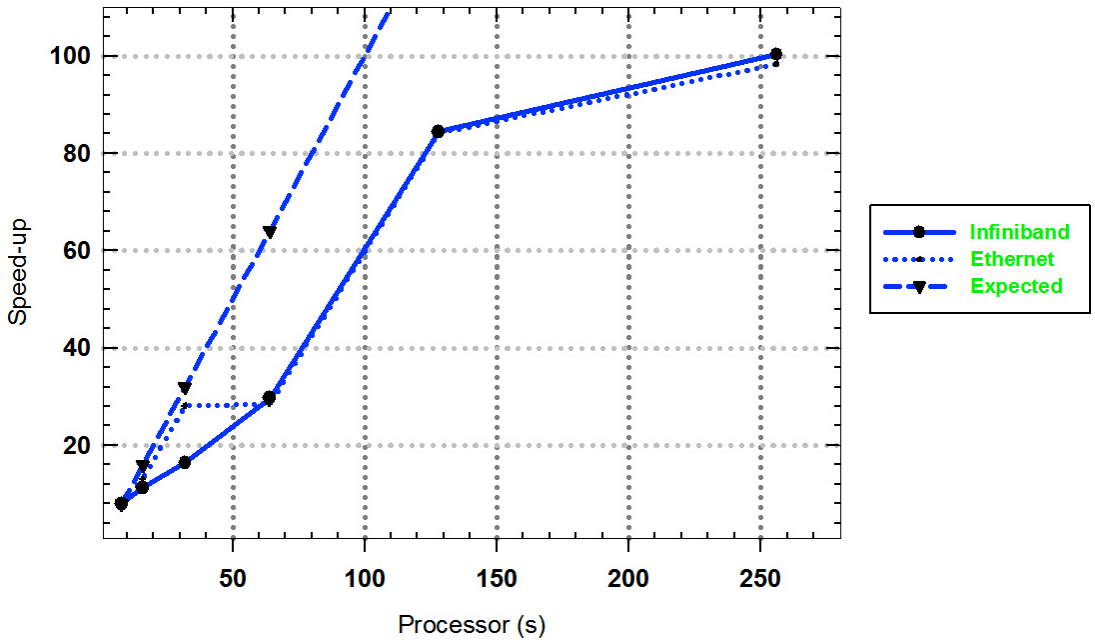

Figure 3: Weak scalability of DL_POLY_2.18

dicate that the performance is comparable to ideal scaling from 4 to 16 processors and suddenly decreases from around 32 to 512 processors. It could be that the method of parallelisation used by DL_POLY_3.09 on the e1350 did not implement very well. A further possibility is that parallel I/O in DL_POLY_3.09 was not efficiently implemented, whereas significant improvements were made in subsequent versions. To this end, different versions of DL_POLY (2.18 and 3.09) were used to simulate the large disilicate glass composition (69120 atoms) in order to analyse performance when using different numbers of processors within the e1350, Sun and Lengau clusters. For this task, all simulations were parallelised using MPI, which distributed computational tasks through the Infiniband network of the clusters. The purpose of this exercise was to understand the effects of the architecture and different choices of DL_POLY versions.

The results (Figure 7) demonstrate the performance of different DL_POLY versions (namely, 2.18, 3.09, 4.07 and Classic_1.9) simulations of the large disilicate glass system (69 210 atoms) on the e1350, Sun and Lengau supercomputers. It indicates that DL_POLY_Classic_1.9 shows essentially ideal performance from 24 to 48 processors and performance slightly decreases from 96 to 144 processors on CHPC's Lengau Petaflop system whose nodes each comprise of 24 processors. However, DL_POLY_Classic_1.9 performed well closer to the ideal value when using 96 and 144 processors of the Lengau cluster. DL_POLY_Classic_1.9 performed well on Lengau probably due to the large number of powerful processors on each node and extensive memory (128 GB) on these compute nodes.

https://doi.org/10.18489/sacj.v29i3.405

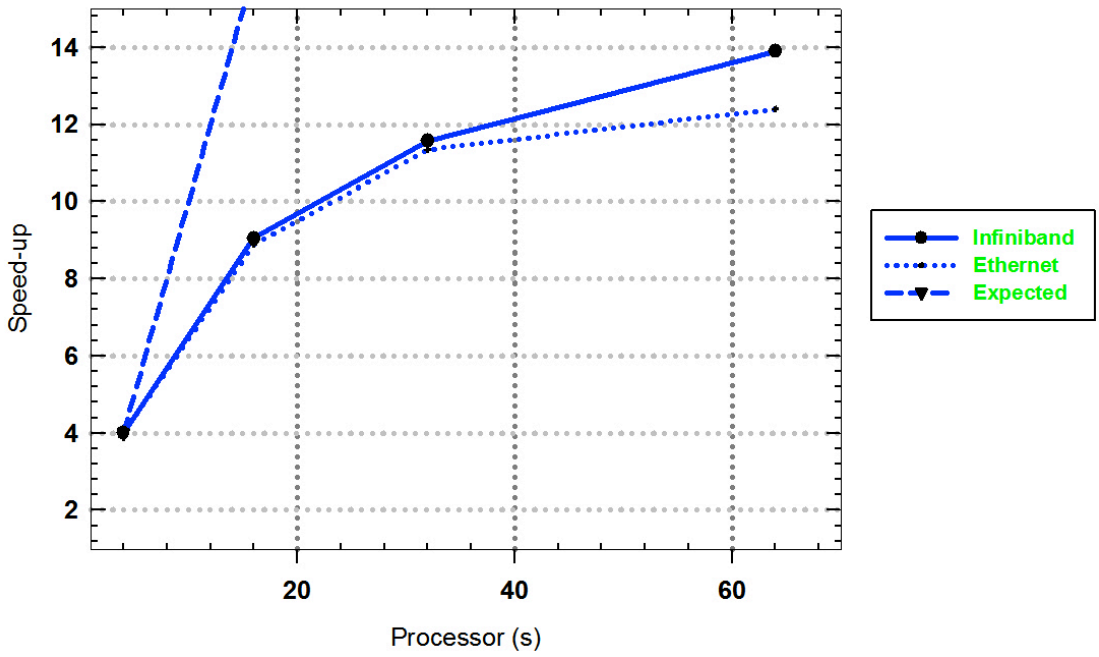

Figure 4: Strong scalability of DL_POLY_3.09 (small system)

However, DL_POLY_4.07 performs reasonably well from 24 to 48 processors, while its performance decreases significantly from 64 up to 144 processors on the Lengau cluster. The gradual reduction in speed of this application from 64 to 144 processors could be caused by large amount of message passing communications between nodes and too little work for each node to do. DL_POLY_2.18 performed close to ideally for 8 to 128 processors on Nehalem nodes of the Sun cluster. The application did, however, not perform well from 8 to 128 processors of the Harpertown architecture on the Sun cluster. This may be attributed to the fact that the Nehalem compute nodes had a larger memory (24GB) than the Harpertown nodes (16GB of memory).

Figure 7 also demonstrates that DL_POLY_3.09 scales poorly compared to ideal scaling for 8 to 128 processors on either the Nehalem or Harpertown nodes of the Sun cluster. On the e1350, DL_POLY_2.18 performs close to the ideal line for 4 to 32 processors and performs quite poorly thereafter. DL_POLY_3.09 performed poorly in terms of scaling on the e1350 in a similar way to what we described above for the Sun cluster. However, the performance of DL_POLY_3.09 slightly increases when one adds more processors on e1350, Nehalem and Harpertown cluster. DL_POLY_3.09 could have performed poorly because it needed more computational resources (processors and memory) on both e1350 and Sun cluster (Nehalem and Harpertown).

In general, performance analyses results (Figure 7) indicate that DL_POLY's latest versions (Classic_1.9 and 4.07) performed reasonably well on Lengau cluster as compared to version 2.18

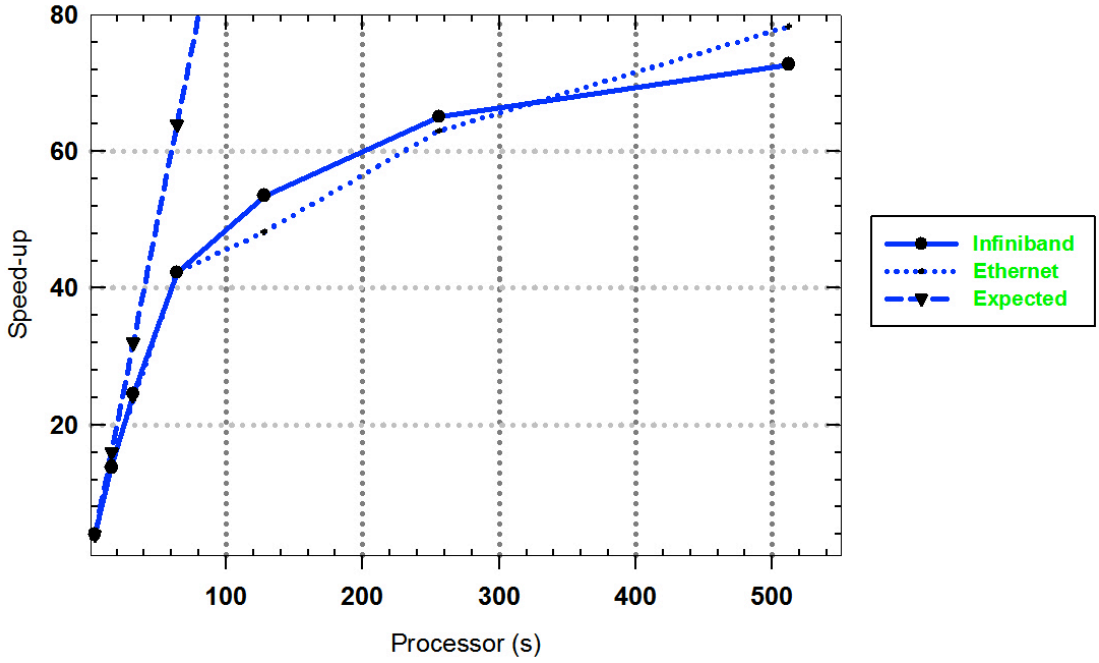

Figure 5: Strong scalability of DL_POLY_3.09 (large simulation)

and 3.09 on both e1350 and Sun cluster. The advanced processors and extensive shared memory within the compute nodes of the Lengau cluster have attributed to the good performance of some versions of DL_POLY. The purpose of this study is to analyse the scaling and relative speed of different DL_POLY versions, of which, the computational speed is significantly higher for Lengau cluster and newer versions of DL_POLY.

4 SUMMARY AND CONCLUSIONS

The findings suggest that scientific users need to understand their problem size in order to select the relevant computational power, and to be aware that using an excessive number of nodes may not necessary increase the performance of the code and could be wasteful of valuable computational resources.

On the problems considered here on the e1350, we see that Ethernet is perfectly good and Infiniband was not needed. However, for bigger systems and better supercomputers such as Sun and Lengau, Infiniband, the modern trend is advantageous – however, we do not have data to compare Ethernet on such clusters as they do not use Ethernet. The DL_POLY_Classic_1.9 code is able to scale well when using a large configuration over Infiniband network even with large increases in the number of nodes within the Lengau supercomputer.

https://doi.org/10.18489/sacj.v29i3.405

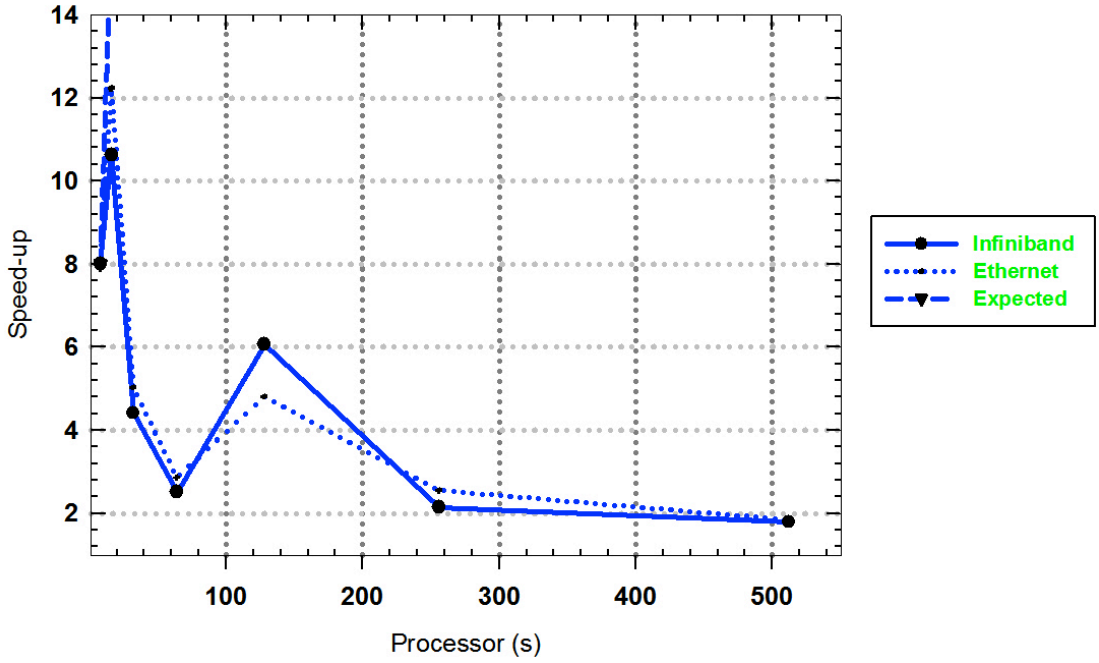

Figure 6: Weak scalability of DL_POLY_3.09

Supercomputing users should ideally utilise DL_POLY_2.18 (as compared to version 3.09) for simulation of small atomic systems on either Infiniband or Ethernet network, depending on the architectural design of the supercomputer. DL_POLY_Classic_1.9 and 4.07 should normally be used to simulate either small or large systems of atoms over the Infiniband network when using many processors within the compute nodes of the supercomputer.

The type of the processors of compute nodes and their memory also play an important role in the overall performance of the parallel application running on a supercomputer. DL_POLY_3.09 could perform well when one utilises a large number of advanced processors possessing large amounts of memory. DL_POLY_Classic_1.9 performed very well on the Lengau cluster which has very powerful compute nodes and large memory compared to the compute nodes of the Sun and e1350 clusters.

It is important for parallel program users to utilise the most advanced available architectures when performing large calculations on supercomputers. To this end, it was found that the latest versions (Classic_1.9 and 4.07) of DL_POLY work better than older versions (2.18 and 3.09) when running large configurations on the supercomputers. It is anticipated that findings of this study will help users of different parallel applications to utilise appropriate computational resources, versions and configurations of the model when performing scientific calculations on the supercomputers.

In future, performance analysis tools (visualisation tools) will be introduced to analyse factors such as network communication, message-passing activities, read/write processes and logic of the

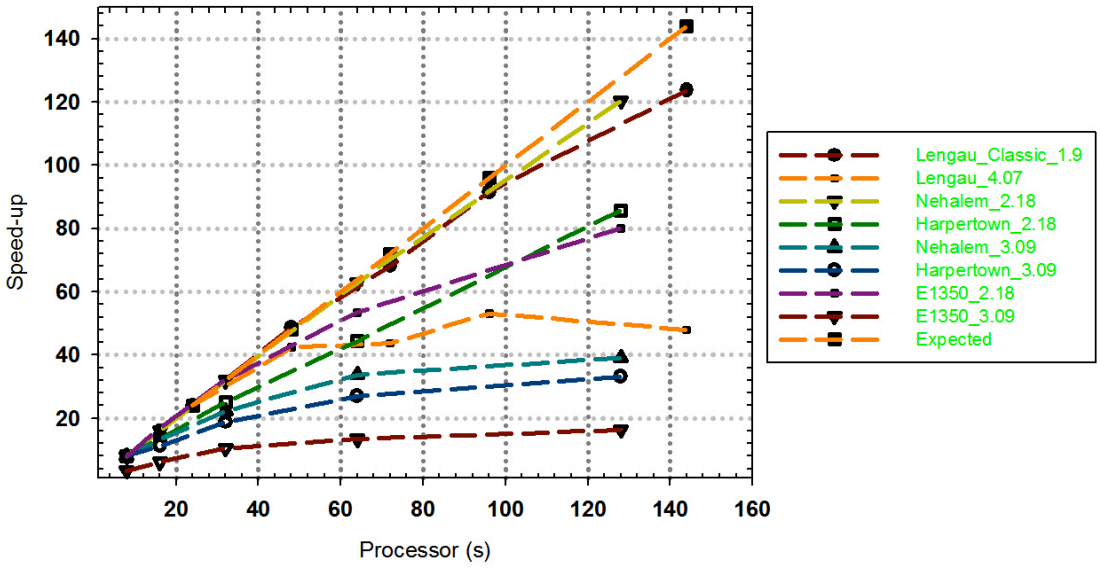

Figure 7: Scaling of DL_POLY on e1350, Sun and Lengau clusters

parallel code executed during the runs.

ACKNOWLEDGEMENTS

This research study is funded by the Centre for High Performance Computing (CHPC); an initiative of the Department of Science and Technology, South Africa in cooperation with the Council for Scientific and Industrial Research (CSIR). We wish to thank the following: CHPC for continual support of this research and providing computational resources to test parallel models. We would also like to thank Prof. Elmarie Biermann for helping to analyse the results of this study.

References

Agarwal, A., Bianchini, R., Chaiken, D., Johnson, K., Kranz, D., Kubiatowicz, J., ... Yeung, D. (1995). The MIT Alewife machine: Architecture and performance. In D. Patterson (Ed.), *ISCA '95 Proceedings of the 22nd annual international symposium on Computer architecture* (pp. 2–13). ACM. https://doi.org/10.1145/223982.223985

Aldrich, E., Fernández-Villaverde, J., Ronald Gallant, A., & Rubio-Ramírez, J. (2011). Tapping the supercomputer under your desk: Solving dynamic equilibrium models with graphics processors. *Journal of Economic Dynamics and Control*, *35*(3), 386–393. https://doi.org/10.1016/j.jedc.2010.10.001

Bertsimas, D., King, A., & Mazumder, R. (2016). Best subset selection via a modern optimization lens. *The annals of statistics*, *44*(2), 813–852. https://doi.org/10.1214/15-AOS1388

Bosilca, G., Delmas, R., Dongarra, J., & Langou, J. (2009). Algorithm-based fault tolerance applied to high performance computing. *Journal of Parallel and Distributed Computing*, *69*(4), 410–416. https://doi.org/10.1016/j.jpdc.2008.12.002

Chamberlain, R., Chace, D., & Patil, A. (1998). How are we doing? An efficiency measure for shared, heterogeneous systems. In O. Bukhres & A. Choudhary (Eds.), *Proceedings of the ISCA 11th international conference on parallel and distributed computing systems* (pp. 15–21). Parallel and Distributed Computing Systems.

Dongarra, J. [J]. (2004). Trends in high performance computing. *The Computer Journal*, *47*(4), 399–403. https://doi.org/10.1093/comjnl/47.4.399

Dongarra, J. [J.], Luszczek, P., & Petitet, A. (2003). The LINPACK benchmark: Past, present, and future. *Concurrency and Computation: Practice and Experience*, *15*(9), https://doi.org/10.1002/cpe.728

Gropp, W. & Snir, M. (2009). On the need for a consortium of capability centres. *International Journal of High Performance Computing Applications*, *23*(4), 413–420. https://doi.org/10.1177/1094342009347706

Heinrich, M., Kuskin, J., Ofelt, D., Heinlein, J., Baxter, J., Pal Singh, J., ... Hennessy, J. (1994). The performance impact of flexibility in the Stanford FLASH multiprocessor. In R. Wexelblat (Ed.), *ASPLOS-VI: Proceedings of the sixth international conference on Architectural support for programming languages and operating systems* (pp. 274–285). https://doi.org/10.1145/195473.195569

Hey, A. (1990). Supercomputing with transputers–Past, present and future. In A. Sameh & H. van der Vorst (Eds.), *ICS '90 Proceedings of the 4th international conference on Supercomputing* (pp. 479–489). ACM. https://doi.org/10.1145/77726.255192

Jiang, D., Shan, H., & Pal Singh, J. (1997). Application restructuring and performance portability on shared virtual memory and hardware-coherent multiprocessors. In M. Berman (Ed.), *PPOPP '97 Proceedings of the sixth ACM SIGPLAN symposium on Principles and practice of parallel programming* (pp. 217–229). ACM. https://doi.org/10.1145/263764.263792

Kepner, J. & Ahalt, S. (2004). Matlabmpi. *Journal of Parallel and Distributed Computing*, *64*(8), 997–1005. https://doi.org/10.1016/j.jpdc.2004.03.018

Kindratenko, V. & Trancoso, P. (2011). Trends in high-performance computing. *Computer in Science and Engineering*, *13*(3), 92–95. https://doi.org/10.1109/MCSE.2011.52

Lange, J., Pedretti, K., Dinda, P., Bridges, P., Bae, C., Soltero, P., & Merritt, A. (2011). Minimal-overhead virtualization of a large scale supercomputer. In E. Pentrank & D. Lea (Eds.), *VEE '11 Proceedings of the 7th ACM SIGPLAN/SIGOPS international conference on Virtual execution environments* (pp. 169–180). ACM. https://doi.org/10.1145/2007477.1952705

Mabakane, M. (2011). *Scaling of scientific software applications on CHPC clustering environment* (Master's thesis, Tshwane University of Technology, Pretoria).

Oyanagi, Y. (2002). Future of supercomputing. *Journal of Computational and Applied Mathematics*, *149*(1), 147–153. https://doi.org/10.1016/S0377-0427(02)00526-5

Smith, W. [W.] & Forester, T. (1996). DL_POLY_2.0: A general-purpose parallel molecular dynamics simulation package. *Journal of Molecular Graphics*, *14*(3), 136–141. https://doi.org/10.1016/S0263-7855(96)00043-4

Smith, W. [W.], Forester, T., & Todorov, I. (2008). The DL_POLY_2 user manual. Last accessed.

Smith, W. [W.] & Todorov, I. (2006). A short description of DL_POLY. *Molecular Simulation*, *32*(12-13), 935–943. https://doi.org/10.1080/08927020600939830

Strohmaier, E., Dongarra, J., Meuer, H., & Simon, H. (2005). Recent trends in the marketplace of high performance computing. *Parallel Computing*, *31*(3+4), 261–273. https://doi.org/10.1016/j.parco.2005.02.001

Tang, E. (2007). *Performance study of a SiO2/water system* (Master's thesis, University of Edinburgh).

Todorov, I. & Smith, W. [W]. (2004). DL_POLY_3: The CCP5 national UK code for molecular-dynamics simulations. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Science*, *362*(1822), 1835–1852. https://doi.org/10.1098/rsta.2004.1419

Todorov, I., Smith, W., Trachenko, K., & Dove, M. (2006). DL_POLY_3: New dimensions in molecular dynamics simulations via massive parallelism. *Journal of Materials Chemistry*, *20*, 1911–1918. https://doi.org/10.1039/B517931A

Varma, J., Wang, C., Mueller, F., Engelmann, C., & Scott, S. (i2006). Scalable, fault tolerant membership for MPI tasks on HPC systems. In G. Egan & Y. Muraoka (Eds.), *ICS '06 Proceedings of the 20th annual international conference on Supercomputing* (pp. 219–228). ACM. https://doi.org/10.1145/1183401.1183433