Solving the Klein-Gordon equation using Fourier spectral methods: A benchmark test for computer performance

S. Aseeri[1], O. Batrašev[2], M. Icardi[3], B. Leu[4], A. Liu[4], N. Li[5], B.K. Muite[2,6], E. Müller[7], B. Palen[8], M. Quell[9], H. Servat[10], P. Sheth[11], R. Speck[12], M. Van Moer[13], J. Vienne[14]

Keywords: Parallel Algorithms; Fast Fourier Transforms; Benchmarks; Partial Differential Equations;

Abstract

The cubic Klein-Gordon equation is a simple but non-trivial partial differential equation whose numerical solution has the main building blocks required for the solution of many other partial differential equations. In this study, the library 2DECOMP&FFT is used in a Fourier spectral scheme to solve the Klein-Gordon equation and strong scaling of the code is examined on thirteen different machines for a problem size of 512^3. The results are useful in assessing likely performance of other parallel fast Fourier transform based programs for solving partial differential equations. The problem is chosen to be large enough to solve on a workstation, yet also of interest to solve quickly on a supercomputer, in particular for parametric studies. Unlike other high performance computing benchmarks, for this problem size, the time to solution will not be improved by simply building a bigger supercomputer.

1. INTRODUCTION

The focusing nonlinear Klein-Gordon equation describes the evolution of a possibly complex scalar field u according to

$$u_{tt} - \Delta u + u = |u|^2 u, \tag{1}$$

where t is time and $\Delta = \partial_{xx} + \partial_{yy} + \partial_{zz}$ the three-dimensional Laplacian. In this note we will focus on real valued functions u only. This equation can exhibit a phenomena known as blow up in which the field u becomes infinite at a certain point. Many partial differential equations which model physical phenomena can exhibit blow up, which typically indicates that the differential equation model is no longer a good model for the physical phenomenon under investigation. Understanding under which initial conditions a blow up in the Klein-Gordon equation will occur is still not clear and it is hoped that parametric simulations can help to elucidate this. Some parametric numerical studies of three dimensional radially symmetric real solutions to the Klein-Gordon equation have recently been done by Donninger and Schlag [2011]. Their parametric study generated a large amount of data, which was unfeasible to store. Their study also made an assumption of spherical symmetry to be able to solve a one-dimensional equation for the radial component of a three-dimensional field. It is of interest to perform similar parametric simulations in three dimensions without symmetry assumptions for the Klein-Gordon and other partial differential equations. Fourier spectral methods are an effective tool for doing this on petascale supercomputers. The

[1]KAUST Supercomputing Laboratory, 4700 King Abdullah University of Science and Technology,Thuwal 23955-6900, Kingdom of Saudi Arabia, [2]Arvutiteaduse instituut, Tartu Ülikool, Liivi 2, Tartu 50409, Estonia, [3]Stochastic Numerics Group, 4700 King Abdullah University of Science and Technology, Thuwal 23955-6900, Kingdom of Saudi Arabia, [4]Electrical and Computer Engineering, University of Michigan, Ann Arbor, MI 48109-2122, USA, [5]Numerical Algorithms Group, Wilkinson House, Jordan Hill Road, Oxford, OX2 8DR, United Kingdom, [6] المملكة العربية السعودية العدَدية، ٤٧٠٠ جَامعة الملك عبدَالله لِلعلوم وَالتقنية، ثول ٢٣٩٥٥-٦٩٠٠، مجموعة الريَاضيَات (Numerical Mathematics Group, 4700 King Abdullah University of Science and Technology, Thuwal 23955-6900, Kingdom of Saudi Arabia), [7]Department of Mathematical Sciences, University of Bath, Bath BA2 7AY, United Kingdom, [8]CAEN Advanced Computing, University of Michigan, Ann Arbor, MI 48109-2094, USA, [9]Institut für Analysis und Scientific Computing, Technische Universität Wien, Wiedner Haupstrasse 8-10, 1040 Wien, Austria, [10]Computer Sciences Department, BSC, Department of Computer Architecture, UPC, c/Jordi Girona, 31 - 08034 - Barcelona, Catalunya, Spain, [11]Biomedical Engineering, University of Michigan, Ann Arbor, MI 48109-2122, USA, [12]Juelich Supercomputing Centre, Forschungszentrum Juelich GmbH, Germany, [13]National Center for Supercomputing Applications, University of Illinois at Urbana-Champaign, Urbana, IL 61801, USA, [14]Texas Advanced Computing Center, University of Texas at Austin, Austin, Texas 78758-4497, USA.

purpose of this paper is to conduct strong scaling experiments of a full model problem on different platforms, trying to understand how much faster a moderate size problem can be made to run. The Klein-Gordon equation is a simple model application for which different choices of numerical methods and computer architectures can be tried to determine which one will give an accurate enough solution at either the fastest time, lowest computational or lowest energy cost. To limit the scope of this study, parallel fast Fourier transforms (FFT) are done with the library 2DECOMP&FFT [Li and Laizet, 2010][2DE, 2014] which primarily uses MPI_ALL_TO_ALLV or MPI_ALL_TO_ALL for its communications. However, it would be interesting to perform a similar study with other parallel Fourier transform libraries (for example PFFT [Pippig, 2013][PFF, 2014], OpenFFT [Duy and Ozaki, 2014][Ope, 2014], PKUFFT [Chen et al., 2010][PKU, 2014] and P3DFFT [Pekurovsky, 2012][P3D, 2014]). The current focus of the study is on a discretization of 512^3 since (i) there are still a wide variety of numerical experiments that can be done for problems of this size, (ii) post processing can be done locally on workstations and (iii) for parametric studies, it is unclear whether access to high-performance computers or a distributed computing cloud based solution is most appropriate. It would also be interesting to try out other numerical methods, such as higher-order time stepping, a lattice-Boltzmann method [Li et al., 2011] and an implicit finite difference/finite element spatial discretization approach with multigrid, fast multipole, tree code and/or preconditioned conjugate gradient solvers. For related work see Gholami et al. [2014], from which it is clear that a careful choice of benchmarking solution is needed when comparing different discretization methods and elliptic system solvers. A later study will describe how such model could be used to rank computing systems as a possible addition to the Linpack benchmark [Meuer et al., 2014] and as an update of the NAS parallel benchmarks [Bailey, 2011]. There are already several attempts to generate benchmarks to supplement the Linpack ranking of supercomputers, such as high performance conjugate gradient [HPC, 2014] and high performance multigrid [HPG, 2014]. However, neither of these solve a real world problem or allow for algorithmic improvements in methods for solving differential equations and linear systems (see for example [Ballard et al., 2014] and for Poisson's equation [Demmel, 1997, p. 277], [Demmel, 2014, Lecture21]). This paper aims to start a discussion on a simple method to rank supercomputers for solving realistic problems that also allows for algorithmic improvements, emphasizes the entire computer eco-system, including software that can be used and developed for that system, and the people that operate the system. Ramachandran et al. [2013] have already noted that better ways to evaluate accelerators are required. By stating a more general problem than solution of a linear system of equations, one can use the best architecture specific algorithm on each platform thereby making a computer ranking not only a measurement of CPU double precision floating point power, but also of problem solving effectiveness.

1.1. The Klein-Gordon Equation

Nakanishi and Schlag [2011] give an introduction to some of the theory of the Klein-Gordon equation, focusing primarily on the three-dimensional radial case. Two-dimensional simulations of the Klein-Gordon equation can be found in Bao and Yang [2007] and Yang [2006]. The linear Klein-Gordon equation occurs as a modification of the linear Schrödinger equation that is consistent with special relativity, see for example Landau [1996] or Grennier [1984]. At the present time, there have been no numerical studies of blow up of solutions to this equation without the assumption of radial symmetry. This equation has generated a large mathematical literature and yet is still poorly understood. Most of this mathematical literature has focused on analyzing the equation on an infinite three-dimensional space with initial data that either decays exponentially as one tends to infinity or is nonzero on a finite set of the domain. Here, we will simulate this equation in a periodic setting. Since this equation is a wave equation, it has a finite speed of propagation of information, much as a sound wave in air takes time to move from one point to another. Consequently for short time simulations, a simulation of a solution that is only nonzero on a finite part of the domain is similar to a simulation on an infinite domain. However, over long times, the solution can spread out and interact with itself on a periodic domain, whereas on an infinite domain, the interaction over long times is significantly reduced and the solution primarily spreads out. Understanding the interactions in a periodic setting is an interesting mathematical problem. Sufficiently smooth solutions of the Klein-Gordon equation conserve the energy given by

$$E(u,u_t) = \int \frac{1}{2}|u_t|^2 + \frac{1}{2}|u|^2 + \frac{1}{2}|\nabla u|^2 - \frac{1}{4}|u|^4 \,\mathrm{d}x.$$

When accurate time stepping schemes are used for sufficiently bounded and differentiable solutions, the energy can act as a test of the correctness of the code implementation, the libraries used and of the computer hardware, since if the energy is not conserved and the implementation is correct, there is likely to be a hardware or library error.

1.1.1. Numerical Schemes

The two time stepping schemes that were used by Donninger and Schlag [2011] for radially symmetric solutions, can be readily adapted to fully three-dimensional simulations using Fourier pseudospectral discretization instead of a finite difference spatial discretization. One of these schemes is simple to implement, and a modification of this for implicit time

stepping is described below since it is typical of numerical methods used for finding approximate solutions to partial differential equations and can be modified for use with other grid based spatial discretization methods.

1.1.2. A Second-Order Scheme

A modification of a second-order scheme used by Donninger and Schlag [2011] and implemented in this study is

$$\frac{u^{n+1}-2u^n+u^{n-1}}{\delta t^2}-\Delta\frac{u^{n+1}+2u^n+u^{n-1}}{4} + \frac{u^{n+1}+2u^n+u^{n-1}}{4}=|u^n|^2u^n, \quad (2)$$

where $u^n \approx u(n\delta t,x,y,z)$. Time stepping takes place in Fourier space where the linear elliptic equation from the semi-implicit time discretization is easy to solve, and the three dimensional Fourier transform is used to obtain the nonlinear term in real space. A more detailed explanation of the method can be found in Cloutier et al. [2012], Rigge [2012] and Balakrishanan et al. [2014]. Example Matlab and Python implementations of this method, as well as the parallel code can be found in Balakrishanan et al. [2014]. This scheme requires two FFTs per time step. The implementation used in this study allows for the field u to be complex.

2. RESULTS

We compare the scalability of the numerical scheme, without output to disk[1]. In all cases the wall clock time for 30 time steps was measured, taken. Figure 1 shows strong scaling results and Table 1 lists the computers according to the shortest run time, as well as documenting the properties of each computer.

2.1. Performance Model

Before discussing the results, it is helpful to have a model for how fast parallel computation can make this computation. Williams et al. [2009] introduced the roofline model which provides a simple upper bound for performance of an algorithm at the node level. Fast Fourier transforms are typically bandwidth limited, the loop based computations in the code are performed on long vectors with unit stride accesses and little re-use of values loaded from memory, and so are similarly bandwidth limited at the node level. For each Fourier Transform, two MPI_ALL_TO_ALL_V or MPI_ALL_TO_ALL calls are used. For small message sizes, these are usually latency limited, while for large message sizes, these are usually network bandwidth limited. Since the problem size considered here is not too large, a simple model would use on chip bandwidth and network latency to determine an estimate of performance and scalability. Let a single core have a bandwidth of B_c, and the minimum network latency for a single byte be L_n. For a discretization of size N^3 grid points, each time step requires approximately d_1N^3 double precision floating point operations and $2d_2[N\log(N)]^3$ operations for the FFT where d_1 and d_2 are constants. Hence, on a single core, the runtime is approximately given by $\frac{d_1N^3+2d_2[N\log(N)]^3}{B_c}$, so that on p processes the runtime is $\frac{d_1N^3+2d_2[N\log(N)]^3}{B_cp}$, assuming no network communication costs. When there is network latency, we get $\frac{d_1N^3+2d_2[N\log(N)]^3}{B_cp}+L_n$. Thus, the network latency gives a lower bound on the possible speedup. It also takes time to send messages across a network. This depends on the topology of the network being used and the algorithm used to perform the MPI_ALL_TO_ALL_V or MPI_ALL_TO_ALL exchange Bertsekas et al. [1991], Grama et al. [2003, p. 537], Swarztrauber [1987], Swarztrauber and Hammond [2001]. Since the problem size is not too large, it is reasonable to assume some small network dependent constant d_3 multiplied by the natural logarithm of the total number of processes (the lower bound for a hypercube network that is well suited to the FFT) so that we obtain $\frac{d_1N^3+d_2[N\log(N)]^3}{B_cp}+L_n+d_3\log(p)$. The model is similar though not as detailed as the ones in Ayala and Wang [2013], Kerbyson and Barker [2011] and Kerbyson et al. [2013]. Since different runtime FFT algorithms on each machine are used, and since the slab decomposition is used for a small number of process and the pencil decomposition is used for more than 512 processes, and since there are different algorithms for performing an all to all exchange, the best of which will depend on the size of the problem being solved, the computer being used and the number and location of the processors being used on that computer (see Foster and Worley [1997]), a more detailed model is not developed. For small p and fixed N, the runtime decreases close to linearly, and once p is large enough the runtime starts to increase again due to communication costs. Since most of the computers used in this study are not hypercubes, the model can only provide a lower bound of the time to solution without communication and computation overlap. Given the large number of computers used, and the fact that even on a single computer, process placement, software environment configuration, as well as communications by other compute jobs would affect program performance, it is infeasible to give a more complete model for the communication cost. It should be noted, that such a model may also be applicable to accelerated computers, if there is sufficient network support and a good implementation, see for example Chen et al. [2010], Czechowski et al. [2011], Czechowski et al. [2012], Park et al. [2013], Song and Hollingsworth [2014] and Swarztrauber

[1] Except for the VSC2 for which no noticeable difference in runtime was observed when doing output to disk, and when not doing output to disk.

Table 1. A table showing ranking of speed at which different computers solve the Klein-Gordon equation for a grid size of 512^3. The systems used, their compute chips, interconnect and underlying one dimensional FFT library used by 2DECOMP&FFT, theoretical chip bandwidth from RAM and theoretical peak floating point performance are also shown. Beacon, Stampede and Titan have accelerators which were not used in the study, hence their properties are not listed nor used in the calculation of theoretical peak double precision floating point performance.

Rank	Machine Name	Time (s)	Cores used	Manufacturer and Model	Node Type	Total Cores	Interconnect	1D FFT Library	Chip Bandwidth Gb/s	Theoretical Peak TFLOP/s
1	Hornet [Hor, 2014]	0.319	12,288	Cray XC40	212 core Intel Xeon 2.5 GHz E5-2680v3	94,656	Cray Aries	FFTW 3 [FFT, 2014]	68	3,784
2	Juqueen [Juq, 2014]	0.350	262,144	IBM Blue Gene/Q	16 core 1.6 GHz Power PC A2	458,752	IBM 5D torus	ESSL [ESS, 2014]	42.6	5,872
3	Stampede [Sta, 2014]	0.581	8,162	Dell Power Edge	28 core Intel Xeon 2.7 GHz E5-2680	462,462	FDR infiniband	Intel MKL [MKL, 2014]	51.2	2,210
4	Shaheen [Sha, 2014]	1.66	16,384	IBM Blue Gene/P	4 core 0.85 GHz PowerPC 450	65,536	IBM 3D torus	ESSL [ESS, 2014]	13.6	222.8
5	MareNostrum III[Mar, 2014]	4.00	64	IBM DataPlex	28 core Intel Xeon 2.6 GHz E5-2670	48,384	FDR10 infiniband	Intel MKL [MKL, 2014]	51.2	1,017
6	Hector [Hec, 2014]	7.66	1024	Cray XE6	216 core AMD Opteron 2.3 GHz 6276 16C	90,112	Cray Gemini	ACML [ACM, 2014]	85	829.0
7	VSC2 [VSC, 2014]	9.03	1024	Megware	28 core AMD Opteron 2.2 GHz 6132HE	21,024	QDR infiniband	FFTW 3 [FFT, 2014]	85	185.0
8	Beacon [Bea, 2014]	9.13	256	Appro	28 core Intel Xeon 2.6 GHz E5-2670	768	FDR infiniband	Intel MKL [MKL, 2014]	51.2	16.0
9	Monte Rosa [Mon, 2014]	11.9	1,024	Cray XE6	216 core AMD Opteron 2.1 GHz 6272	47,872	Cray Gemini	ACML [ACM, 2014]	85	402.1
10	Titan [Tit, 2014]	17.0	256	Cray XK7	16 core AMD Opteron 2.2 GHz 6274	299,008	Cray Gemini	ACML [ACM, 2014]	85	2,631
11	Vedur [Ved, 2014]	18.6	1,024	HP ProLiant DL165 G7	216 core AMD Opteron 2.3 GHz 6276	2,560	QDR infiniband	FFTW 3 [FFT, 2014]	85	236
12	Aquila [Aqu, 2014]	22.4	256	ClusterVision	24 core Intel Xeon 2.8 GHz E5462	800	DDR infiniband	FFTW 3 [FFT, 2014]	12.8	8.96
13	Neser [Nes, 2014]	27.9	128	IBM System X3550	24 core Intel Xeon 2.5 GHz E5420	1,024	Gigabit ethernet	FFTW 3 [FFT, 2014]	10.7	10.2

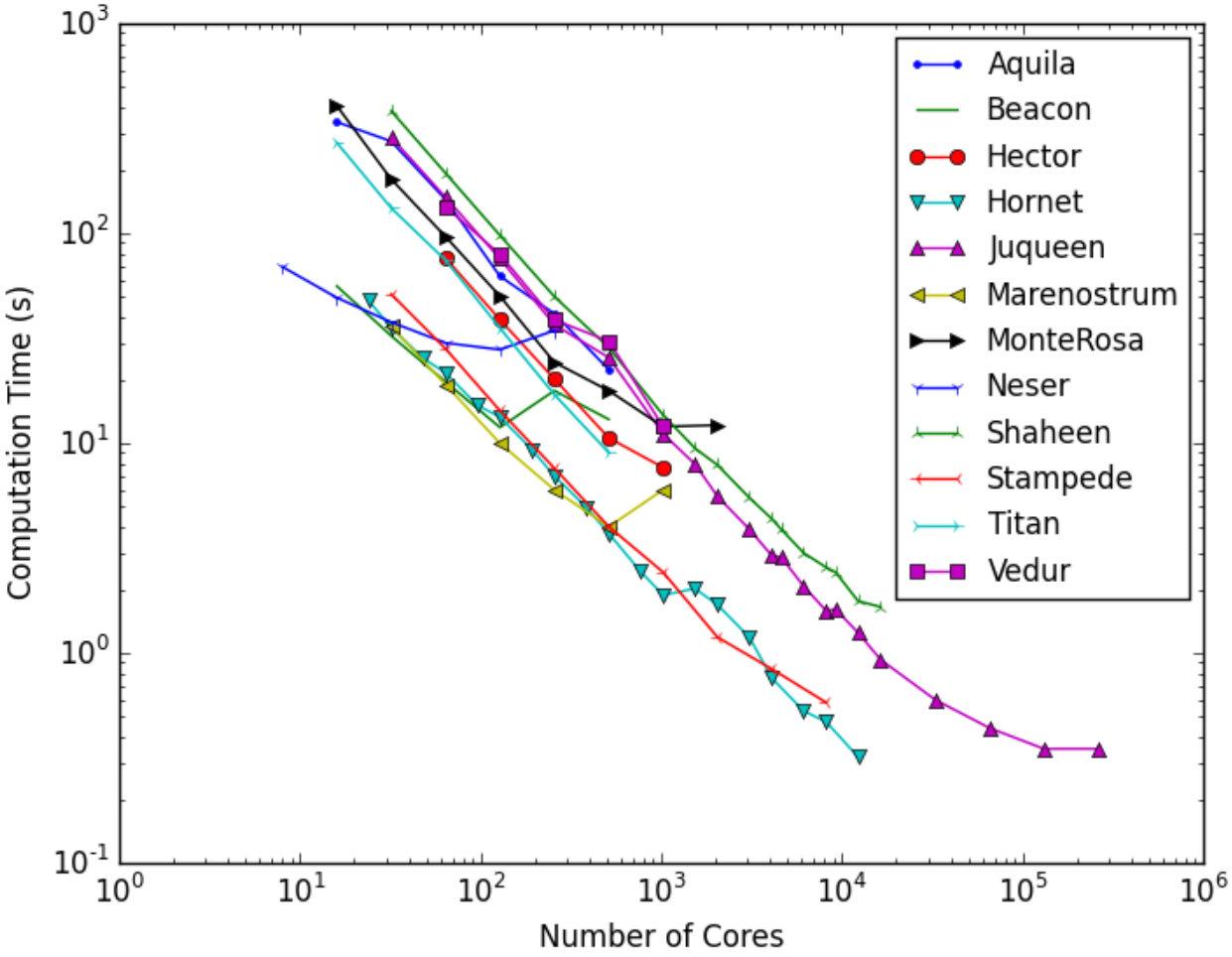

Figure 1. Scaling results showing computation time for 30 time steps as a function of the number of processor cores.

[1997]. The model does not account for overlap of computation and communication in computing a distributed Fourier transform, which have been tried by Kandalla et al. [2011] and Hoefler and Zerah [2007] (MPI 3.0 non blocking collectives make this easier to implement), for which one might be able to decrease the lower bound. The work here is similar in spirit to that in Worley and Foster [1994], but the equation used is simpler, and hence easier to use for evaluating computer performance.

2.2. Result Summary

Table 1 shows that for this problem, a ranking can be obtained that would allow for easy evaluation of computer architectures for solving partial differential equations of a given discretization in the fastest time possible using algorithms that are dominated by Fourier transforms. Figure 1 shows that the strong scaling limit is not reached on all the platforms primarily due to clusters being too small or due to queue size restrictions. For cases where less cores were used than close to full system size and the strong scaling limit was not reached, further computation time is required, in particular on Titan and Hornet. It is likely that the results in the table can change with extensive tuning and careful job placement, however they are representative of what occurs in a typical production environment. Figure 2 shows scaling, but rather than using core count, uses total processor bandwidth which is defined as bandwidth per node multiplied by number of nodes used. Czechowski et al. [2012] and Marjanovic et al. [2014] indicate that node level bandwidth may be more important than system interconnect band-

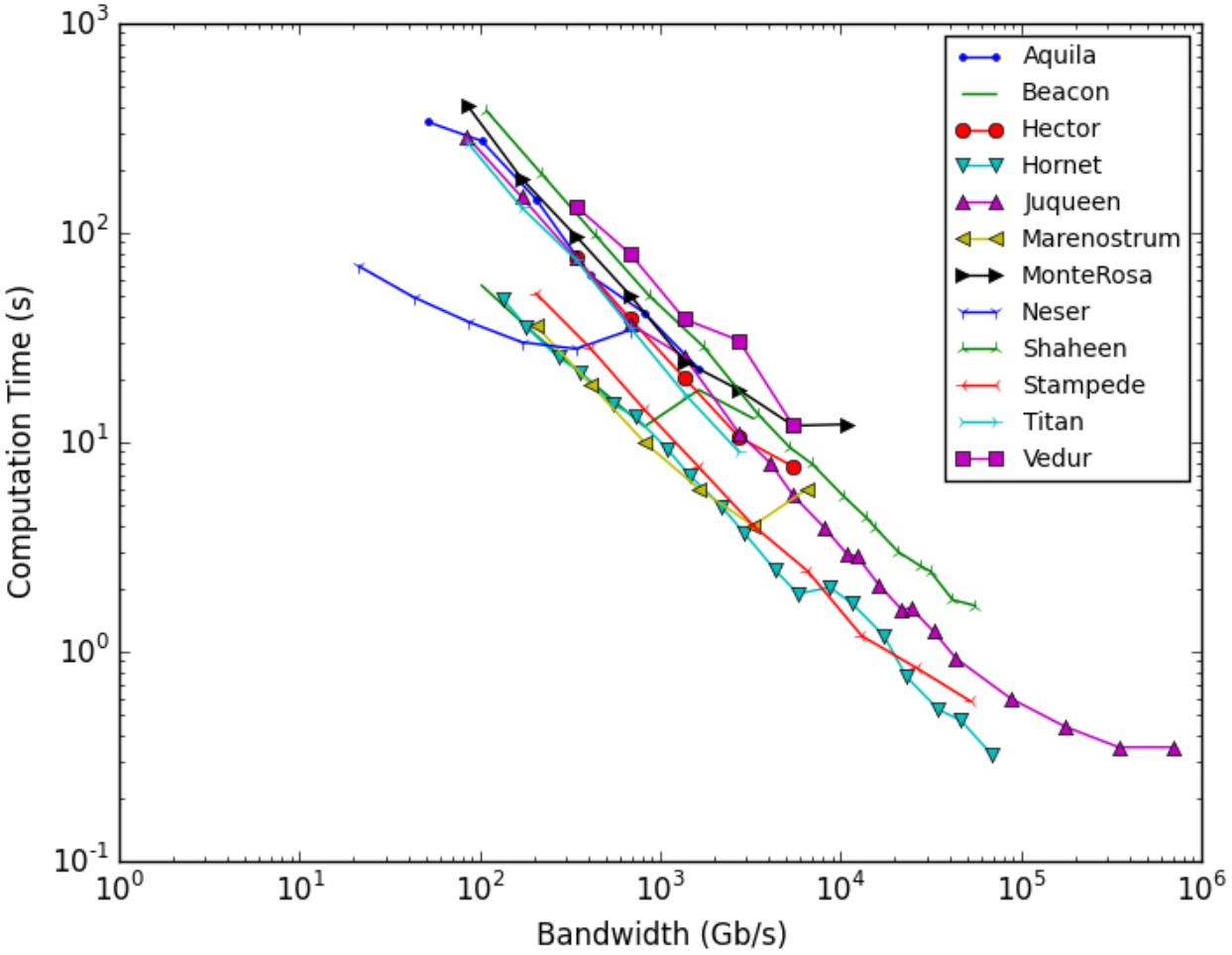

Figure 2. Scaling results showing computation time for 30 time steps as a function of total on chip bandwidth defined as the maximum theoretical bandwidth from RAM on a node multiplied by the number of nodes used.

width, and indicate that for benchmarks which look at a fixed problem size, a few key performance indicators can replace the benchmark – for the current code, memory bandwidth and performance of MPI_ALL_TO_ALL are good indicators and for HPCG[HPC, 2014], Marjanovic et al. [2014] indicate that memory bandwidth and MPI_ALL_REDUCE time are good performance indicators. Good performance for MPI_ALL_TO_ALL is much harder to achieve than good performance for MPI_ALL_REDUCE and will in particular imply good performance for MPI_ALL_REDUCE, though likely at a higher hardware cost.

2.3. Result Discussion

Table 1 shows that Hornet produces the fastest run time due to its high performance communication network and fast processors. Juqueen does however have a much higher theoretical peak floating point performance than Hornet. Juqueen's large number of processor cores are difficult to use efficiently for this problem size, thus despite Hornet's smaller size, it is more effective at solving the Klein Gordon equation using the current algorithm. Similar behavior is observed on Marenostrum III, where network performance makes it difficult to utilize a large portion of the machine despite the high theoretical peak performance. On Aquila, there is a pronounced drop in performance when going from 8 to 16 cores, and after this the scaling is close to ideal again relative to the 16 core run (and

the efficiency for 256 cores would be 76.5% if it was measured relative to the 16 core results). This drop in performance is likely because the 8 core run only requires intra-node communications, whereas all runs with higher core counts have to send messages via the slower infiniband interconnect. A similar drop in performance is observed on Hornet in going from 2048 cores to 3072 cores, since MPI communication requires two communication steps on the dragon fly topology rather than just one communication step. On Beacon, the runtime was quite sensitive to process placement, likely due to network topology. Neser is the oldest machine used in this study. It is a Linux cluster with a 1Gb ethernet network that is still fast at small core counts. Neser's low cost gigabit ethernet network prevents good scaling behavior to larger core counts and explains its low ranking in table 1. Speed up on Shaheen was very close to ideal due to the fast interconnect and balanced design which allows for good throughput to the cores given their maximum floating point performance. Shaheen has 0.85 GHz cores and Juqueen has cores that are clocked at 1.6GHz and can do twice as many floating point operations per cycle, yet for the same number of cores, Juqueen is on average only 1.4 times faster than Shaheen, this is likely because each core on Juqueen has a lower share of bandwidth (2.66Gb/s) than on Shaheen (3.375 Gb/s). The performance on Vedur is significantly worse than on Hector, despite having the same compute chips. This is likely due to interconnect latency. Finally, fig. 2 also shows that newer Intel processors on Beacon, Hornet, Marenostrum and Stampede give much better performance than the Power PC, older Intel and AMD processors for the same node level bandwidth. The reason for the improved performance is likely to be due to the larger number of floating point operations that can be done per cycle compared to the other architectures, and the more sophisticated memory controllers – this indicates that simply knowing chip bandwidth and performance of MPI_ALL_TO_ALL allows for a simple but incomplete model. As explained by Lo et al. [2014], these characteristics of a machine are also difficult to measure precisely so their use for predictions should be done with care, though they may give a good but not perfect initial approximation.

3. FUTURE WORK

The results in this paper give a guide for codes which heavily utilize the Fourier transform on the different combinations of processor and interconnect that will give the best overall computation time. There are a variety of other methods for solving the same equation, and other aspects of using supercomputers that have not been covered in the present study, but would be useful to cover in other studies. Possible future work includes: finite difference/finite element codes using multigrid, tree code, fast multipole or conjugate gradient linear equation solvers for the implicit linear system solve in the time discretized Klein-Gordon equation, high order timestepping, in-situ visualization, measurement of computer energy consumption, use of accelerators, effectiveness of input and output, and more detailed performance models can be done in further work on a smaller set of computers. Care will be required in choosing initial conditions to allow for a meaningful comparison with other ways of discretizing this equation and solving the linear equations in implicit time stepping schemes.

Acknowledgments.

The authors thankfully acknowledge the computer resources, technical expertise and assistance provided by: The Beacon Project at the University of Tennessee supported by the National Science Foundation under Grant Number 1137097; The UK's national high-performance computing service, which is provided by UoE HPCx Ltd at the University of Edinburgh, Cray Inc and NAG Ltd, and funded by the Office of Science and Technology through EPSRC's High End Computing Programme; The Barcelona Supercomputing Center - Centro Nacional de Supercomputación; The Swiss National Supercomputing Centre (CSCS); The Texas Advanced Computing Center (TACC) at The University of Texas at Austin; The KAUST Supercomputer Laboratory (KSL) at King Abdullah University of Science and Technology (KAUST) for providing the resources that have contributed to this study; The Oak Ridge Leadership Computing Facility at the Oak Ridge National Laboratory, which is supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC05-00OR22725; The Aquila HPC service at the University of Bath; The Vienna Scientific Cluster (VSC); The PRACE research infrastructure resources in Germany at HLRS and FZ Juelich; The High Performance Computing Center of the University of Tartu. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the funding bodies or the service providers.

Initial research to start this project used resources of Kraken at the National Institute for Computational Science, Trestles at the San Diego Supercomputing Center both through the Extreme Science and Engineering Discovery Environment (XSEDE), which is supported by National Science Foundation grant number ACI-1053575, the University of Michigan High Performance Computing Service FLUX and Mira at the Argonne Leadership Computing Facility at Argonne National Laboratory, which is supported by the Office of Science of the U.S. Department of Energy under contract DE-AC02-06CH11357

Oleg Batrašev and Benson Muite were partially supported by the Estonian Centre of Excellence in Computer Science (EXCS) under the auspices of the European Regional De-

velopment Funds. Mark Van Moer's contribution was made possible through the XSEDE Extended Collaborative Support Service (ECSS) program. We also thank Stefan Andersson, Winfried Auzinger, Andy Bauer, David E DeMarle, David Ketcheson, David Keyes, Robert Krasny, Dmitry Pekurovsky, Michael Pippig, Paul Rigge, Madhusudhanan Srinivasan, Eero Vainikko, Matthias Winkel, Brian Wylie, Hong Yi and Rio Yokota for helpful advice and suggestions.

REFERENCES

2decomp&fft, November 2014. URL http://2decomp.org/.

AMD Core Math Library, November 2014. URL http://developer.amd.com/tools-and-sdks/cpu-development/amd-core-math-library-acml/.

Aquila cluster, November 2014. URL https://wiki.bath.ac.uk/display/HPC/Aquila.

Beacon cluster, November 2014. URL https://www.nics.tennessee.edu/beacon.

IBM Engineering and Scientific Subroutine Library., November 2014. URL http://www-03.ibm.com/systems/power/software/essl/.

Fast Fourier Transform in the West., November 2014. URL http://fftw.org/.

High performance conjugate gradient, November 2014. URL https://software.sandia.gov/hpcg/html/index.html.

High performance geometric multigrid., November 2014. URL https://hpgmg.org/.

Hector supercomputer, November 2014. URL http://www.hector.ac.uk/.

Hornet supercomputer, November 2014. URL http://www.hlrs.de/systems/platforms/cray-xc40-hornet/.

Juqueen supercomputer, November 2014. URL http://www.fz-juelich.de/ias/jsc/juqueen.

Intel math kernel library., November 2014. URL https://software.intel.com/en-us/intel-mkl/.

Marenostrum supercomputer, November 2014. URL http://www.bsc.es/marenostrum-support-services.

Monte rosa supercomputer, November 2014. URL http://www.cscs.ch/computers/monte_rosa.

Neser cluster, November 2014. URL http://www.hpc.kaust.edu.sa/documentation/user_guide/resources/neser/.

Openfft, November 2014. URL http://www.openmx-square.org/openfft/.

P3dfft, November 2014. URL https://code.google.com/p/p3dfft/.

Pfft, November 2014. URL https://www-user.tu-chemnitz.de/~mpip/software.php#pfft.

Pkufft, November 2014. URL https://code.google.com/p/parray-programming/wiki/Chapter_7_Case_Studies.

Shaheen, November 2014. URL http://ksl.kaust.edu.sa/Pages/Shaheen.aspx.

Stampede supercomputer, November 2014. URL https://www.tacc.utexas.edu/user-services/user-guides/stampede-user-guide.

Titan supercomputer., November 2014. URL https://www.olcf.ornl.gov/computing-resources/titan-cray-xk7/.

VSC2 cluster., November 2014. URL http://vsc.ac.at/about-vsc/vsc-pool/vsc-2/.

Vedur cluster., November 2014. URL http://www.hpc.ut.ee/en/vedur_cluster.

O. Ayala and L.-P Wang. Parallel implementation and scalability analysis of 3D fast Fourier transform using 2D domain decomposition. *Parallel Computing*, 39:58–77, 2013.

D.H. Bailey. *Encyclopedia of Parallel Computing*, chapter NAS Parallel Benchmarks., pages 1254–1259. Springer, 2011.

S. Balakrishanan, A.H. Bargash, G. Chen, B. Cloutier, N. Li, B.K. Muite, M. Quell, P. Rigge, M. Solimani, A. Souza, A.S. Thiban, J. West, D. Malicke, M. Van Moer, and D. San Roman Alerigi. Parallel spectral numerical methods., November 2014. URL http://en.wikibooks.org/w/index.php?title=Parallel_Spectral_Numerical_Methods.

G. Ballard, E. Carson, J. Demmel, M. Hoemmen, N. Knight, and O. Schwartz. Communication lower bounds and optimal algorithms for numerical linear algebra. *Acta Numerica*, pages 1–155, 2014.

W.Z. Bao and L. Yang. Efficient and accurate numerical methods for the Klein-Gordon-Schrodinger equations. *J. Comp. Phys.*, 225(2):1863–1893, 2007.

D.P. Bertsekas, C. Özveren, G.D. Stamoulis, P. Tseng, and J.N. Tsitsiklis. Optimal communication algorithms for hypercubes. *Journal of Parallel and Distributed Computing*, 11:263–275, 1991.

Y. Chen, X. Cui, and H. Mei. Large-scale FFT on GPU clusters. In *Proc. International Conference on Supercomputing*, 2010.

B. Cloutier, B.K. Muite, and P. Rigge. Performance of FORTRAN and C GPU extensions for a benchmark suite of Fourier pseudospectral algorithms. In *Proc. Symposium on Application Accelerators ih High Performance Computing (SAAHPC)*, pages 1145–1148, 2012.

K. Czechowski, C. McClanahan, C. Battaglino, K. Iyer, P.-K. Yeung, and R. Vuduc. Prospects for scalable 3D FFTs on heterogeneous exascale systems. In *Proc. ACM/IEEE Conf. Supercomputing (SC)*, 2011.

K. Czechowski, C. McClanahan, C. Battaglino, K. Iyer, P.-K. Yeung, and R. Vuduc. On the communication complexity of 3D FFTs and its implications for exascale. In *Proc. ACM Int'l. Conf. Supercomputing (ICS)*, 2012.

J. Demmel. Applications of parallel computing. Course lecture notes, 2014. URL `http://www.cs.berkeley.edu/~demmel/cs267_Spr14/`.

J.W. Demmel. *Applied Numerical Linear Algebra*. SIAM, 1997.

R. Donninger and W. Schlag. Numerical study of the blowup/global existence dichotomy for the focusing cubic nonlinear Klein-Gordon equation. *Nonlinearity*, 24:2547–2562, 2011.

T.V.T. Duy and T. Ozaki. A decomposition method with minimum communication amount for parallelization of multi-dimensional FFTs. *Computer Physics Communications*, 185:153–164, 2014.

I.T. Foster and P.H. Worley. Parallel algorithms for the spectral transform method. *SIAM J. Sci. Stat. Comput.*, 18(3): 806–837, 1997.

A. Gholami, D. Malhotra, H. Sundar, and G. Biros. FFT, FFM or multi grid? A comparative study of state-of-the-art Poisson solvers. arXiv:1408.6497, 2014.

A. Grama, A. Gupta, G. Karypis, and V. Kumar. *Introduction to Parallel Computing.* Addison Wesley, 2nd edition, 2003.

W. Grennier. *Relativistic Quantum Mechanics.* Springer, 1984.

T. Hoefler and G. Zerah. Optimization of a parallel 3D-FFT with non-blocking collective operations. 3rd International ABINIT Developer Workshop, 2007. URL `http://htor.inf.ethz.ch/publications/index.php?pub=44`.

K. Kandalla, H. Subramoni, K. Tomko, D. Pekurovsky, S. Sur, and D.K. Panda. High-performance and scalable non-blocking all-to-all with collective offload on infiniband clusters: a study with parallel 3D FFT. *Comput. Sci Res Dev*, 26:237–246, 2011.

D.J. Kerbyson and K.J. Barker. A performance model of direct numerical simulation for analyzing large-scale systems. In *Proc. Workshop on Large Scale Parallel Processing, IEEE International Parallel and Distributed Processing*, 2011.

D.J. Kerbyson, K.J. Barker, D.S. Gallo, D. Chen, J.R. Brunheroto, K.D. Ryu, G.L. Chiu, and A. Hoisie. Tracking the performance evolution of Blue Gene Systems. In *Proc. ISC 13, Lecture Notes in Computer Science*, pages 317–329. Springer Verlag, 2013.

R.H. Landau. *Quantum Mechanics II.* Wiley, 1996.

N. Li and S. Laizet. 2DECOMP&FFT - A highly scalable 2D decomposition library and FFT interface. In *Proc. Cray User Group*, 2010.

Q. Li, Z. Ji, Z. Zheng, and H. Liu. Numerical solution of nonlinear Klein-Gordon equation using Lattice Boltzmann method. *Applied Mathematics*, 2:1479–1485, 2011.

Y.J. Lo, S. Williams, B. Straalen, T. Ligocki, M. Cordery, L. Oliker, M. Hall, and N. Wright. Roofline model toolkit: A practical tool for architectural and program analysis. In *Proc. 5th International Workshop on Performance Modeling, Benchmarking and Simulation of High Performance Computer Systems (PMBS14)*, 2014.

V. Marjanovic, J. Garcia, and C. Glass. Performance modeling of the hpcg benchmark. In *Proc. 5th International Workshop on Performance Modeling, Benchmarking and Simulation of High Performance Computer Systems (PMBS14)*, 2014.

H. Meuer, E. Strohmaier, J. Dongarra, H. Simon, and M. Meuer. Top 500 list, November 2014. URL `http://www.top500.org/`.

K. Nakanishi and W. Schlag. *Invariant Manifolds and Dispersive Hamiltonian Evolution Equations.* European Mathematical Society, 2011.

J. Park, G. Bikshandi, K. Vaidyanathan, P.T.P. Tang, P. Dubey, and D. Kim. Tera-scale 1D FFT with low-communication algorithm and Intel Xeon®Ph™ coprocessors. In *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis (SC '13)*, number 34, 2013.

D. Pekurovsky. P3DFFT: A framework for parallel computations of Fourier transforms in three dimensions. *SIAM J. Sci. Comput.*, 34:C192–C209, 2012.

M. Pippig. PFFT: An extension of FFTW to massively parallel architectures. *SIAM J. Sci. Comput.*, 35(3):C213–C236, 2013.

A. Ramachandran, J. Vienne, R. Vand Der Wijngaart, L. Koesterke, and I. Sharapov. Performance evaluation of NAS parallel benchmarks on Intel Xeon Phi. In *Proc. 42nd International Conference on Parallel Processing*, 2013.

P. Rigge. Numerical solutions to the Sine-Gordon equation. arXiv:1212.2716, 2012.

S. Song and J.K. Hollingsworth. Designing and auto-tuning parallel 3-D FFT for computation-communication overlap. In *Proc. of the 19th ACM SIGPLAN symposium on Principles and practice of parallel programming*, pages 181–192, 2014.

P.N. Swarztrauber. Multiprocessor FFTs. *Parallel Comput.*, 5:197–210, 1987.

P.N. Swarztrauber. Multipipeline multiprocessor system. US 5689722 A, 1997. URL `http://www.google.com/patents/US5689722`.

P.N. Swarztrauber and S.W. Hammond. A comparison of optimal FFTs on torus and hypercube multicomputers. *Parallel Computing*, 27:847–859, 2001.

S. Williams, A. Waterman, and D. Patterson. Roofline: an insightful visual performance model for multicore architectures. *Communications of the ACM*, 53(4):65–76, 2009.

P. Worley and I.T. Foster. Parallel spectral transform shallow water model: a runtime-tunable parallel benchmark code. In D.W. Dongarra, J.J. Walker, editor, *Proc. of the Scalable High Performance Computing Conference*, pages 207–214, 1994.

L. Yang. Numerical studies of the Klein-Gordon-Schrödinger equations. Master's thesis, National University of Singapore, 2006. URL `http://scholarbank.nus.edu.sg/handle/10635/15515`.

Biography

Samar Aseeri (email: `samar.asseeri@kaust.edu.sa`) is a computational scientist at the King Abdullah University of Science and Technology.

Oleg Batrašev (email: `olegus@ut.ee`) is a researcher in computer science at Tartu Ülikool.

Matteo Icardi (email: `matteo.icardi@kaust.edu.sa`) is a postdoctoral research fellow at the King Abdullah University of Science and Technology.

Brian Leu (email: `brianleu@umich.edu`) is an undergraduate student in electrical engineering at the University of Michigan.

Albert Liu (email: `alberliu@umich.edu`) is an undergraduate student in electrical engineering and physics at the University of Michigan.

Ning Li (email: `ning.li@nag.co.uk`) is a high performance computing consultant at the Numerical Algorithms Group. Ning specializes in numerical software development, in particular in the area of high performance computing and wrote the parallel software framework 2DECOMP&FFT.

Benson Muite (email: `benson.muite@ut.ee`) is a postdoctoral research fellow in computer science at Tartu Ülikool.

Eike Müller (email: `e.mueller@bath.ac.uk`) is a postdoctoral research associate in mathematics at the University of Bath.

Brock Palen (email: `brockp@umich.edu`) is a high performance computing system administrator at the University of Michigan.

Michael Quell (email: `michael.quell@yahoo.de`) is an undergraduate student in mathematics at the Technische Universität Wien.

Harald Servat (email: `harald.servat@bsc.es`) is a computer science researcher at the Barcelona Supercomputing Center.

Parth Sheth (email: `pssheth@umich.edu`) is a research assistant in biomedical engineering at the University of Michigan.

Robert Speck (email: `r.speck@fz-juelich.de`) is a mathematician at Forschungszentrum Juelich, Juelich Supercomputing Centre.

Mark Van Moer (email: `mvanmoer@illinois.edu`) is a visualization scientist at the National Center for Supercomputing Applications.

Jerome Vienne (email: `viennej@tacc.utexas.edu`) is a computational scientist at the Texas Advanced Computational Center.