FORSCHUNGSZENTRUM JÜLICH GmbH
Jülich Supercomputing Centre
D-52425 Jülich, Tel. (02461) 61-6402

Technical Report

Jülich Blue Gene/P
Extreme Scaling Workshop 2010

Bernd Mohr, Wolfgang Frings (Eds.)

FZJ-JSC-IB-2010-03

May 2010

(last change: 17-May-2010)

Jülich Blue Gene/P Extreme Scaling Workshop 2010

Bernd Mohr and Wolfgang Frings
Jülich Supercomputing Centre

Executive Summary

From 22 to 24 March, JSC organized the 2010 edition of its Blue Gene Scaling Workshop. Like the last workshop in October 2009[1], the main focus were application codes able to scale-up during the workshop to the full Blue Gene/P system JUGENE which consists of 72 racks with a total of 294,912 cores – still the highest number of cores worldwide available in a single system.

Interested application teams had to submit short proposals which were evaluated with respect to the required extreme scaling, application-related constraints which had to be fulfilled by the JUGENE software infrastructure and the scientific impact that the codes could produce. Ten high-quality applications were selected[1]:

- Towards Direct Numerical Simulation of a Billion Fully Resolved Rigid Bodies Immersed in a Fluid
  J. Götz, **K. Iglberger**, M. Stürmer, U. Rüde
  University of Erlangen-Nuremberg, Germany

- Simulation of Fluid Flow and Mass Transport at Extreme Scale
  S. Khirevich, **A. Daneyko**, U. Tallarek
  Department of Chemistry, Philipps University of Marburg, Germany

- Highly Resolved Simulations of Turbulent Flows in Complex Geometries with the YALES2 Solver
  V. Moureau, P. Domingo, L. Vervisch
  CORIA, Université et INSA de Rouen, France

- Scalability of the Nek5000 Spectral Element Code
  S. Kerkemeier, **S. Parker**, P. Fischer
  ETH Zurich, Switzerland and Argonne National Laboratory, USA

[1] Participants to the workshop marked in **bold**

- Full Scale Simulation of Coronary Arteries in Presence of Red Blood Cells
  A. Peters, S. Melchionna, E. Kaxiras, J. Lätt, M. Bernaschi, M. Bisson, S. Succi
  Harvard University, USA and EFPL, Switzerland, and
  Consiglio Nazionale delle Ricerche, Italy

- Scaling Parallel Fast Fourier Transform on BlueGene/P
  M. Pippig, D. Potts
  Chemnitz University of Technology, Germany

- Extreme Scaling of the BQCD Benchmark
  H. Stüben, **M. Allalen**
  ZIB Berlin and LRZ Munich, Germany

- MP^{2}C – A Parallel Mesoscopic Particle Dynamics Program
  G. Sutmann
  Jülich Supercomputing Centre, Germany

- Hydrodynamic Turbulence Induced by Sedimenting Particles
  M. Uhlmann
  Karlsruhe Institute of Technology, Germany

- KKRnano: A Program for Large-Scale Density-Functional Calculations
  R. Zeller, A. Thiess
  IAS-3 and IFF-1, Forschungszentrum Jülich, Germany

During the workshop, the teams were supported by JSC parallel application experts, the JUGENE system administrators and one IBM MPI expert; however, the participants shared a lot of expertise and knowledge, too. More than half of the teams succeeded to submit one or more successful full 72 rack jobs during the course of the workshop, and three more, where algorithmic or load balancing issues required the codes to run on a power-of-two number of cores, scaled their applications to 64 racks (262,144 cores). One team "only" achieved to run on 32 racks (131,072 cores) due to a program bug which could not be resolved during the short time of the workshop. A total of 392 jobs were launched using 138.72 rack days of the total 164 rack days reserved for the workshop. This is an 84% utilization which could only be reached because of the extreme good stability of the system and the proactive maintenance of JSC and IBM staff.

Achieved Results

Godehard Sutmann from the Jülich Supercomputing Centre, already a participant of the last workshop[1], executed a few experiments on the full machine to get beyond the results achieved last time where due to using a distribution of 2^n domains "only" 262,144 cores could be used. Also, various processor mappings were tested to improve a communication bottleneck.

The following chapters give an account on more detailed execution and scaling results achieved by the other application codes during the workshop provided by the participants themselves.

Acknowledgements

We would like to thank IBM Germany for their support of the workshop.

References

[1] Bernd Mohr, Wolfgang Frings, Jülich Blue Gene/P Extreme Scaling Workshop 2009, Technical Report FZJ-JSC-IB-2010-02, Forschungszentrum Jülich, February 2010.
`http://www.fz-juelich.de/jsc/docs/autoren2010/mohr1/`

Towards Direct Numerical Simulation of a Billion Fully Resolved Rigid Bodies Immersed in a Fluid

Jan Götz, Klaus Iglberger, Markus Stürmer, Ulrich Rüde
Chair for System Simulation, University Erlangen-Nuremberg

Abstract

This report describes computational models for particle-laden flows based on a fully resolved fluid structure interaction. The flow simulation uses the Lattice Boltzmann method, while the particles are handled by a rigid body dynamics algorithm. The particles can have individual non-spherical shapes, creating the need for a non-trivial collision detection and special contact models. An explicit coupling algorithm transfers momenta from the fluid to the particles in each time step, while the particles impose moving boundaries for the flow solver. All algorithms and their interaction are fully parallelized. Scaling experiments and a careful performance analysis are presented for up to 294 912 processor cores of the Jugene. The largest simulations involve 264 million particles that are coupled to a fluid which is simultaneously resolved by 150 billion cells for the Lattice Boltzmann method.

Introduction

Simulations of particulate flows are crucial for the modeling of many natural phenomena and for the optimization of related industrial applications. Sedimentation and fluidization processes are important examples. Many of the currently established simulation methods, for instance molecular dynamics or particle hydrodynamics, do not resolve the particles in the flow, but treat them as point masses without explicitly accounting for individual frictional collisions.

In our approach, a 3D lattice Boltzmann (LBM) fluid simulation [9, 2] and a rigid body dynamics [1] simulation are dynamically coupled in order to fully resolve the motion of immersed particles. Both programs are fully parallelized and coupled via an explicit exchange of momenta from the fluid onto the objects [6, 7], and by modeling moving boundaries on the surface of the objects to transfer momenta back to the fluid [10]. Further details on the rigid body dynamics solver and the LBM solver are described in [4] and [3], more details on the coupling can be found in [5].

Performance Results

In this chapter we present performance and scalability results for the JUGENE with up to 294 912 processor cores.
To compile the program, the GNU C and C++ compiler in version 4.1.2 is used. Tests with the standard IBM XLC compiler showed a lower performance compared to the GNU compiler when compiling with flag -O2. Higher optimization was not possible due to internal compiler errors[1].
In order to compare the performance values, we present the results in terms of *million lattice updates per second* (MLUPS). We evaluate two different scenarios, one labeled case A with sparsely packed particles, which appears in particulate flows and one case B with densely packed particles representative for sedimentation or segregation processes.

Node Performance of coupled Fluid Structure Interaction Simulations

We evaluate the node performance of the coupled fluid particle interaction simulations, which will obviously depend on the domain size per core and the number of objects. The results are shown in Table 1, where the GFlop/s and the memory bandwidth values are again measured using the automatically available performance counters [8].
For the smaller domain size of 40^3 lattice cells per core, the performance is in general lower than for domain size of 80^3 lattice cells due to a larger communication to computation ratio, although the performance of the pure lattice Boltzmann solver is higher for a domain size of 40^3 lattice cells per core. When the number of rigid bodies is increased (case B), the computations for mapping the objects to the grid, the force evaluation, the movement and the collisions of the objects are higher compared to simulations with smaller number of objects (case A), resulting in a lower performance in terms of MLUPS for case B. Thus, the highest MLUPS value is obtained for a domain size of 80^3 lattice cells and case A. For the same domain size and the same test case, also the highest measured values for GFlop/s and main memory bandwidth are achieved.

Test case	Lattice domain size per core	MLUPS	% of peak GFlop/s performance	% of peak memory bandwidth
A	40^3	3.0	4.12	14.61
	80^3	3.24	**4.59**	**22.32**
B	40^3	1.9	3.0	11.18
	80^3	1.92	3.15	15.63

Table 1: Node performance of coupled fluid structure interaction simulations.

[1]For higher optimization the XLC returned "1586-494 (U) INTERNAL COMPILER ERROR" due to problems with variable length array functions. This information was reported to IBM.

Multi Node Performance of coupled Fluid Structure Interaction Simulations

The weak scaling is performed with both scenarios and two different domain sizes. The experiment is started at 64 cores with a decomposition of $4 \times 4 \times 4$, when 8 processes communicate in all neighboring directions. Note that placement of MPI processes needs to be done according to the requested torus shape and thus the underlying network shape on the system to maintain a high performance.

The weak scaling up to 294 912 cores of case A and case B is presented in Figure 1. For case A, the reference values for calculating the efficiency are 44.07 MLUPS for a domain size of 40^3 lattice cells per core and 47.31 MLUPS for a domain size of 80^3 lattice cells per core for measurements with 64 cores. The lower efficiency for the smaller domain size results from a higher communication to computation ratio. The efficiency for case B is based on measurements on

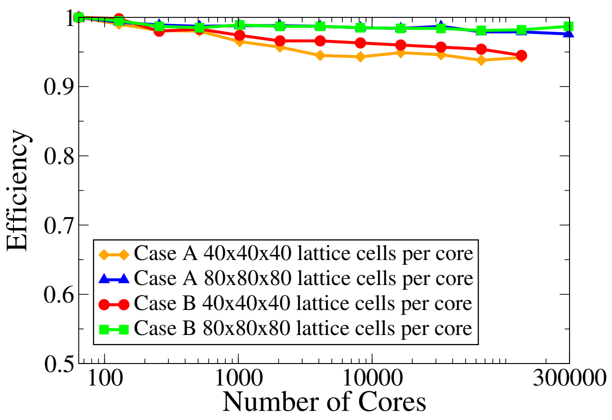

Figure 1: Weak scaling from 64 to 294 912 compute cores for sparsely packed (case A) and densely packed particles (case B).

64 cores, which result in 23.06 MLUPS and 25,14 MLUPS for domain sizes of 40^3 and 80^3 lattice cells per core, respectively. Over the whole range of cores, the efficiency remains over 98% for a domain sizes of 80^3 lattice sells per core. Again, the efficiency of the simulation with smaller domain size is lower, compared to the simulation with larger domain size due to the communication to computation ratio.

Figure 2 illustrates a simulation of a real world scenario. Objects with density values of $0.8\, kg/dm^3$ and $1.2\, kg/dm^3$ are immersed in water with density $1\, kg/dm^3$ and a gravitation field. Thereby light objects will rise to the top, while heavy objects will fall to the ground.

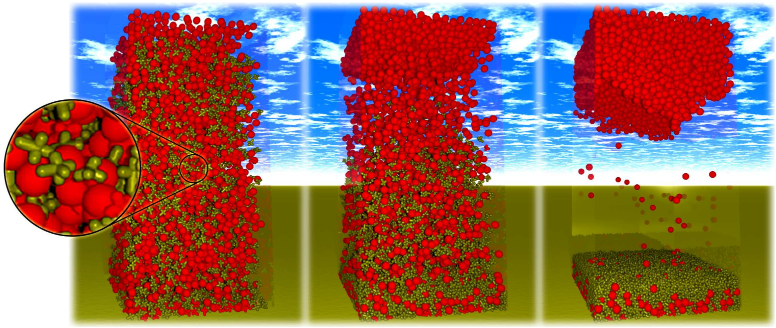

Figure 2: Segregation simulation of 12 013 objects with two different shapes in different time steps simulated on 2 048 cores in a box. Density values of $0.8\,kg/dm^3$ and $1.2\,kg/dm^3$ are used for the objects in water with density $1\,kg/dm^3$ and a gravitation field. Light particles are rising to the top of the box, while heavy particle fall to the bottom.

Conclusion

We have developed an efficient simulation system for the direct numerical simulation of particulate flows built from a combination of a lattice Boltzmann fluid simulation and a rigid body physics engine. The results of the workshop for the weak scaling on the Jugene Blue Gene/P system demonstrate a good parallel efficiency on the full machine with 294 912 cores. The performance also confirmed the extension of the code to a pure local treatment of the rigid body solver. This enables a near-perfect scaling on the full machine. However, because of long runtimes when writing some output information on all nodes, we disabled all output besides the timing information for the benchmarking runs. Still, some of our full machine runs did not complete due to errors caused by exceeded MPI buffers. We are investigating the problem, but still are not sure if these are triggered by software errors in our code, or hardware problems of the machine.

Acknowledgments: We would like to thank Bernd Mohr, Wolfgang Frings and Lukas Arnold from Jüelich Supercomputing Centre (JSC) for their assistance, support and discussions. Also the generous allocation of compute time on the Jugene system is gratefully acknowledged.

References

[1] M. Anitescu. Optimization-based simulation of nonsmooth rigid multibody dynamics. *Math. Program.*, 105(1):113–143, 2006.

[2] S. Chen and G. D. Doolen. Lattice Boltzmann method for fluid flows. *Annu. Rev. Fluid Mech.*, 30:329–364, 1998.

[3] C. Feichtinger, J. Götz, S. Donath, K. Iglberger, and U. Rüde. Concepts of waLBerla prototype 0.1. Technical Report 07–10, Computer Science Department 10 (System Simulation), University of Erlangen-Nuremberg, 2007.

[4] K. Iglberger and U. Rüde. Massively parallel rigid body dynamics simulation. *Comp. Sci. - Res. Dev.*, 23(3):159, 2009.

[5] K. Iglberger, N. Thürey, and U. Rüde. Simulation of moving particles in 3D with the Lattice Boltzmann method. *Comput. Math. Appl.*, 55(7):1461–1468, 2008.

[6] A. J. C. Ladd. Numerical simulations of particulate suspensions via a discretized Boltzmann equation. Part 1. Theoretical foundation. *J. Fluid Mech.*, 271:285–309, 1994.

[7] A. J. C. Ladd. Numerical simulations of particulate suspensions via a discretized Boltzmann equation. Part 2. Numerical results. *J. Fluid Mech.*, 271:311–339, 1994.

[8] G. Lakner, I-H. Chung, G. Cong, S. Fadden, N. Goracke D. Klepacki, J. Lien, C. Pospiech, S. R. Seelam, and H.-F. Wen. IBM System Blue Gene Solution: Performance Analysis Tools. pages 29–35, 2008. IBM order number REDP-4256-01.

[9] S. Succi. *The Lattice Boltzmann Equation - For Fluid Dynamics and Beyond*. Clarendon Press, 2001.

[10] D. Yu, R. Mei, L.-S. Luo, and W. Shyy. Viscous flow computations with the method of lattice Boltzmann equation. *Prog. Aerosp. Sci.*, 39(5):329–367, 2003.

Simulation of Fluid Flow and Mass Transport at Extreme Scale

Siarhei Khirevich, Anton Daneyko, and Ulrich Tallarek
Department of Chemistry, Philipps University of Marburg

Description of the Code

Flow and mass transport in porous media play a central role in a variety of analytical, industrial, and natural processes, including chromatographic separations, reaction engineering, catalysis, soil remediation, and etc. Detailed understanding of transport processes in porous media guides the design strategies and performance for the aforementioned processes. In our study, we address the phenomena of flow and hydrodynamic dispersion in porous media represented by confined random packings of the spherical particles, which are the model of, for instance, packed chromatographic columns [1] or fixed-bed chemical reactors [2].

Our simulation workflow consists of the following steps:

1. *Creation of the random close sphere packing* with the help of the Jodrey-Troy algorithm (JT) [3]. JT starts from random distribution of sphere centers within a simulation domain. Such a distribution introduces overlaps between spheres which are iteratively removed by JT. The algorithm exits when there are no more overlaps exist in the sphere packing.

2. *Spatial discretization of the generated packing.* This step initializes uniform grid and sets up each voxel of the grid to the 'solid' or 'fluid' according to the spatial position of the voxel center (inside or outside of the closest sphere, respectively). We used a spatial resolution of 30 lattice nodes per particle diameter, which is sufficient for the accurate simulation of the fluid flow in the random-close sphere packings [4, 5].

3. *Simulation of the fluid flow* in the packing using the Lattice-Boltzmann method (LBM) [6]. The method simulates flow of Newtonian fluid solving the discrete Boltzmann equation. After the simulation is completed, the resulting fluid flow velocity field is written into a file. Depending on the problem dimensions, the file size can reach 1TB.

4. *Simulation of the advective-diffusive mass transport* using the Random-Walk Particle Tracking method (RWPT) [7] and the fluid velocity filed obtained on the previous simulation step. The idea of RWPT is to displace

(due to advection and diffusion) a large number of inert tracer particles in the volume of interest and to track the displacement of the tracer ensemble over time. The goal of this simulation is to analyze the transient behavior of the hydrodynamic dispersion coefficient and to determine its close-to-asymptotic value.

Packing generation is done on a single processor-core while discretization, flow and mass transport simulations are performed using parallel programs (written in C/MPI). Over 99% of CPU time is spent by programs simulating fluid flow (LBM) and mass transport (RWPT), therefore, we selected LBM and RWPT to be scaled up to the whole JUGENE system.

The time to reach close-to-asymptotic value of the hydrodynamic dispersion coefficient is defined, in particular, by the length scales of spatial heterogeneities (void space variation within a packing) existing in the random porous media [1]. Confined spherical packings have spatial heterogeneities appearing on the scale of the whole transverse dimension of the confining container. As a consequence, we operate with 'long' computational domains, i.e. longitudinal dimension of a domain significantly exceeds (by factor of hundreds) the transverse one. This packing configuration enables the use of the one-dimensional ('slice') decomposition of the computational domain. For the extreme scaling workshop, we prepared a packing with dimensions of $632 \times 632 \times 294912$ lattice nodes reflecting a typical size of the simulation domain. An exact choice of the longest domain dimension (294912 lattice nodes) is arbitrary, the only requirement is the packing (simulation domain) length must be large enough to observe close-to-asymptotic behavior of the hydrodynamic dispersion coefficient [4].

Workshop

At the extreme scaling workshop we benchmarked our programs to estimate strong performance scaling and I/O performance. The selected problem size of $632 \times 632 \times 294912$ lattice nodes resulted in memory requirements of 12TB for LBM and 6TB for RWPT programs. Due to the memory limitations, we selected the performance at 32k and 16k processor cores (8 and 4 BlueGene/P racks) as the baseline for LBM and RWPT, respectively. The strong scaling of LBM and RWPT is shown on the Figure 1a. Both LBM and RWPT demonstrate an identical scaling behavior, which is related to the similar decomposition techniques used in both of the methods and, consequently, similar distribution of fluid nodes (n_{fl}) among processors. In case of LBM, n_{fl} defines the number of lattice links to be processed while for RWPT n_{fl} specifies the number of tracer particles associated with a given processor (assuming uniform concentration of tracer particles in the fluid phase of the packing). Non-uniformity of n_{fl} distribution among processors leads to workload imbalance, and is caused by two factors:

1. The smallest indivisible data chunk for the implemented decomposition procedure and a given problem size is a two dimensional layer with di-

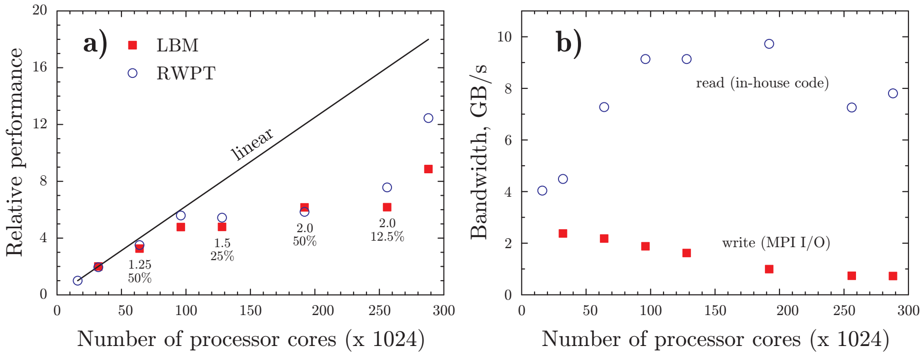

Figure 1: a) Performance scaling on JUGENE system. The upper number in each pair indicates the ratio of the longitudinal dimensions of the longest to the shortest decomposed subdomains. The lower number is the fraction of the processes with the longest domain length. b) I/O performance of LBM (file write) and RWPT (file read).

mensions of $632 \times 632 \times 1$ lattice nodes. The whole simulation domain consists of $L_z = 294912$ silces, which may or may not be evenly divisible by the number of allocated processor cores n. The remainder of the L_z/n slices, if present, is distributed among some of the processes, which yields a non-uniform distribution of n_{fl}.

2. Non-uniform distribution of n_{fl} on the lattice is an inherent property of the random porous media model (for the packing used in simulations, the ratio between maximal $n_{fl,max}$ and minimal $n_{fl,min}$ numbers of fluid nodes per lattice layer is 1.27).

Depending on whether L_z is a multiple of the current processor number n or not, only the second or both factors cause workload imbalance. More efficient scaling of RWPT can be explained by the larger ratio of computation to communication times compared to LBM.

I/O benchmarks (one velocity field file with size of 1.3 TB) were performed using MPI I/O and in-house developed approaches. The latter was implemented because on previously employed BlueGene/P system MPI I/O was not available. During preparations for the extreme scaling workshop we found some issues with file writing on JUGENE file system using our approach, therefore we implemented MPI I/O. The I/O performance results are presented on the Figure 1b. Performance decrease of MPI I/O write with increasing number of allocated cores is probably related to the increasing amount of events when more than one MPI task access the same file system block (2MB) at the same time. Read benchmarks show significant performance increase up to 100k processor cores and then performance stays roughly constant with further increase of allocated cores. The achieved read performance of 9GB/s is approximately 30% of the total file system bandwidth.

Conclusion

The extreme scaling workshop was very helpful for us: results achieved on the workshop demonstrate current performance and limits of both LBM and RWTP. Fruitful discussions with experts available during the workshop, Wolfgang Frings and Pascal Vezolle, resulted in new ideas on further improvements of all aspects of the codes performance (single-core, parallel, and I/O).

References

[1] S. Khirevich, A. Höltzel, A. Seidel-Morgenstern, and U. Tallarek. *Time and length scales of eddy dispersion in chromatographic beds.* Anal. Chem. **81**: 7057-7066 (2009)

[2] H. Freund, J. Bauer, T. Zeiser, and G. Emig. *Detailed simulation of transport processes in fixed-beds.* Ind. Eng. Chem. Res. **44**: 64236434 (2005)

[3] W. S. Jodrey and E. M. Tory. *Computer simulation of close random packing of equal spheres.* Phys. Rev. A **32**: 2347-2351 (1985)

[4] R.S. Maier, D. M. Kroll, R. S. Bernard, S. E. Howington, J. F. Peters, and H. T. Davis. *Pore-scale simulation of dispersion.* Phys. Fluids **12**: 2065-2079 (2000)

[5] R. S. Maier and R. S. Bernard. *Lattice-Boltzmann accuracy in pore-scale flow simulation.* J. Comput. Phys. **229**: 233-255 (2010)

[6] S. Chen and G. D. Doolen. *Lattice boltzmann method for fluid flows.* Annu. Rev. Fluid Mech. **30**: 329-364 (1998)

[7] P. Salamon, D. Fernandez-Garcia, and J. J. Gomez-Hernandez. *A review and numerical assessment of the random walk particle tracking method.* J. Contam. Hydrol. **87**: 277-305 (2006)

Highly Resolved Simulations of Turbulent Flows in Complex Geometries with the YALES2 Solver

V. Moureau, P. Domingo and L. Vervisch
CORIA, CNRS, Université et INSA de Rouen

Description of the Code

YALES2 aims at the solving of two-phase combustion from primary atomization to pollutant prediction on massive complex meshes. It is able to handle efficiently unstructured meshes with several billions of elements, thus enabling the Direct Numerical Simulation of laboratory and semi-industrial configurations. The solvers of YALES2 cover a wide range of phenomena and applications, and may be assembled to address multi-physics problems (Moureau, 2010).

YALES2 solves the low-Mach Navier-Stokes equations with a projection method for constant and variable density flows. These equations are discretized with a 4th-order central scheme in space and a 4th-order Runge-Kutta like scheme in time. The efficiency of projection approaches is usually driven by the performances of the Poisson linear solver. In YALES2, the linear solver is a highly efficient Deflated Preconditioned Conjugate Gradient that has two mesh levels. Another issue in such large computations is the mesh generation. The strategy in YALES2 is to generate meshes sufficiently resolved to describe the geometry with around 50 million elements and then to automatically refine them with non-degenerescent tessellation algorithms (Rivara, 1984). Finally, any CPU memory issues are alleviated thanks to mesh partitioning. A 10 billion cell mesh for instance would consist of at least 10,000 parts written in compressed HDF5 files.

Accomplished Objectives

During the workshop, direct numerical simulation of an industrial swirl burner was performed with a mesh of 21 billion tetrahedral cells. The mesh resolution of 50 μmallowed to have around 1700 points in each direction of the combustor. Results obtained before the workshop with a 2.6 billion tetrahedral-based mesh are shown in Fig. 1. More details about the geometry and its modeling may be found in Moureau *et al.* (2007); Galpin *et al.* (2008); Roux *et al.* (2005); Moureau *et al.* (2010).

The code was able to run during ten iterations with this huge mesh on 32768 compute nodes, i.e. 32 racks, in SMP mode. The

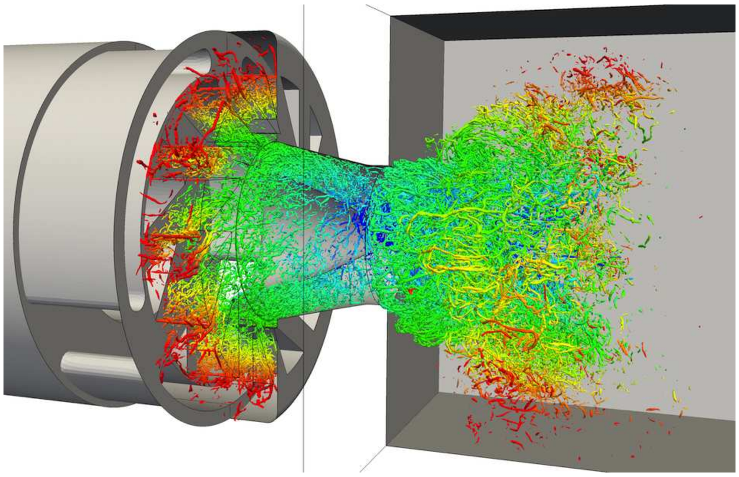

Figure 1: Smallest resolved vortices in the iso-thermal computation of the PRECCINSTA burner with a mesh of 2.6 billion cells.

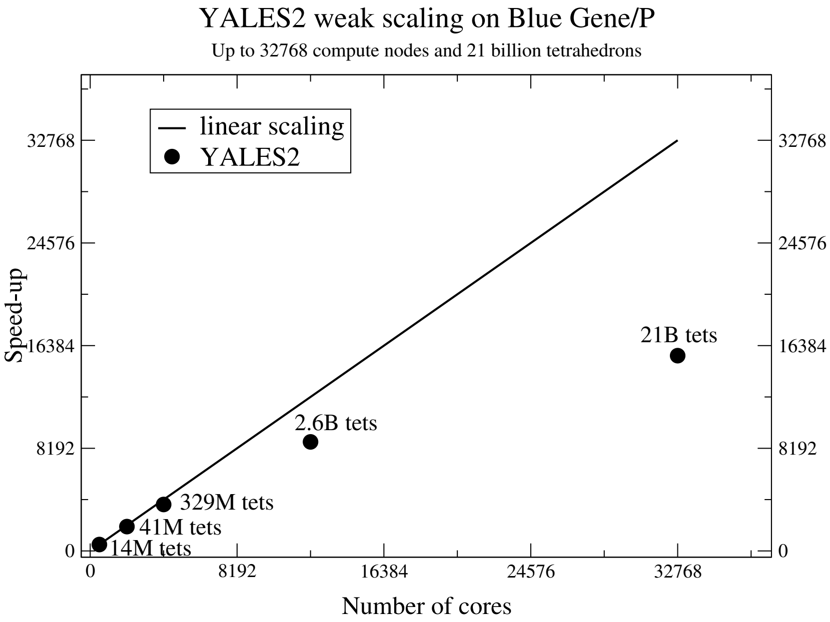

Figure 2: Weak scaling of YALES2 on Jugene

code also ran on 65536 compute nodes, i.e. 64 racks, in SMP and VN modes but the duration of the grid partitioning phase and the limited available runtime prevented to begin the flow solver iterations. During the workshop, the grid partitioning, which was identified as the main bottleneck, was improved in several manners to shorten the initialization phase. For instance, a new multi-block format for pre-partitioned meshes was coded and tested successfully. It consists in storing several mesh blocks in the same mesh partition file. This dual partitioning allows to pre-partition a mesh with 16384 cores in 262144 blocks stored in 16384 files as a first step before running on 64 racks of the machine. From the 10 flow solver iterations of the successful run, the speed-up was measured and it is reported in Fig. 2.

The participation to the workshop was a unique opportunity to highlight the main bottlenecks in the handling of such massive meshes. Grid partitioning and file IO remain the great challenges for future finite-volume flow solvers. The performances measured during the workshop also shows that more than two mesh levels may be required in the solving of the Poisson equation to obtain a better speed-up.

Acknowledgements

The authors would like to acknowledge Bernd Mohr and the team of the Juelich Supercomputing Centre for the workshop organization and their kind support. Some of the runs were prepared on the Blue Gene/P machine at IDRIS in France under the allocation 2009-020152 made by GENCI (Grand Equipement National de Calcul Intensif).

References

Galpin, J., Naudin, A., Vervisch, L., Angelberger, C., Colin, O. & Domingo, P. 2008 Large-eddy simulation of a fuel-lean premixed turbulent swirl-burner. *Combustion and Flame* **155** (1-2), 247 – 266.

Moureau, V. 2010 `nonpremixed.insa-rouen.fr/~moureau/yales2.html`.

Moureau, V., Domingo, P. & Vervisch, L. 2010 Large-eddy simulations and direct numerical simulations of turbulent premixed combustion in an industrial swirl burner. *submitted to Combustion and Flame* .

Moureau, V., Minot, P., Pitsch, H. & Bérat, C. 2007 A ghost-fluid method for large-eddy simulations of premixed combustion in complex geometries. *Journal of Computational Physics* **221** (2), 600 – 614.

Rivara, M.-C. 1984 Mesh refinement processes based on the generalized bisection of simplices. *SIAM Journal on Numerical Analysis* **21** (3), 604–613.

Roux, S., Lartigue, G., Poinsot, T., Meier, U. & Berat, C. 2005 Studies of mean and unsteady flow in a swirled combustor using experiments, acoustic analysis, and large eddy simulations. *Comb. Flame* **141** (1-2), 40–54.

Scalability of the Nek5000 Spectral Element Code

Stefan Kerkemeier, ETH Zurich
Scott Parker, Argonne National Laboratory
Paul Fischer, Argonne National Laboratory

Description of the Code

Nek5000 (http://nek5000.mcs.anl.gov) is an open-source CFD code for the simulation of unsteady incompressible / low Mach number fluid flow, heat transfer, combustion, and MHD in general three-dimensional domains. It features a stabilized formulation that makes it appropriate for direct numerical simulation (DNS) and large eddy simulation (LES) of turbulent flows.

Spatial discretization is based on the spectral element method (SEM), which is a high order weighted residual technique that combines the geometric flexibility of the finite element method (FEM) with the tensor-product efficiency of spectral methods. In the SEM, the solution is approximated by tensor product polynomials of order N on each of E elements, giving rise to $n \approx EN^3$ unknown basis coefficients for each field (velocity, pressure, temperature, etc.). Typical values of N are 8–16, which implies roughly a thousandfold decrease in the number of elements compared to an FEM mesh having the same number of gridpoints, n.

Nek5000 employs high-order semi-implicit timestepping schemes that decouple the Navier-Stokes equations into independent advection, diffusion, and pressure projection substeps. The principal computational bottleneck in simulating unsteady incompressible and low-Mach number flows is the elliptic problem governing the pressure, which must be computed implicitly at each timestep. The SEM-based Poisson system is solved with preconditioned conjugate gradients or GMRES. Preconditioning is based on variational multigrid using local overlapping Schwarz methods for element-based smoothing at resolution N and $\approx N/2$, coupled with a global coarse-grid problem based on linear elements, which is solved in parallel using algebraic multigrid. To efficiently use the Blue Gene/P system architecture tuned computational and communication kernels are implemented.

The code is used by dozens of institutions world-wide. Among other things, it has been used to study long-standing problems of spatio-temporal chaos within Rayleigh-Benard convection problems and transition to turbulence in vascular flows. Nek5000 is currently being used for several large-scale problems, including detailed analysis of rod-bundle flows in next generation nuclear reactors, ocean current modeling, and turbulent autoignition with realistic kinetics and

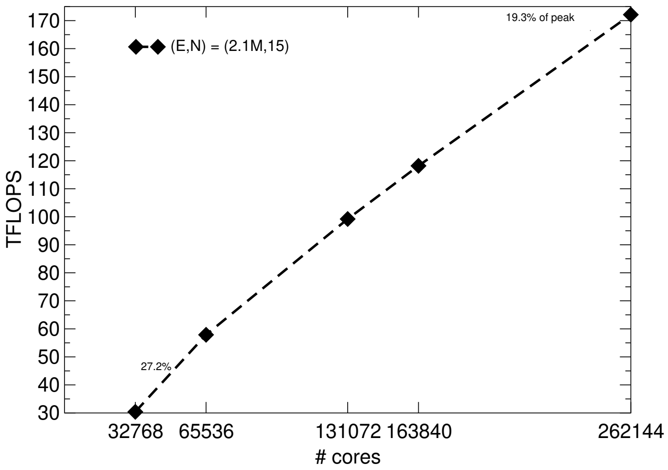

Figure 1: Scaling of Nek5000 on Jugene

transport.

Accomplished Objectives

Nek5000 was successfully run on all 72 racks of the Julich Supercomputing Centre's Blue Gene/P system. Unexpected issues related to non-power-of-two processor partitioning resulted in non-optimal performance at 72 racks, but excellent scaling was observed for up to 262,144 cores (64 racks). The plot above shows strong scaling results for a simulation with a (128 x 128 x 128) spectral element mesh of order N=15 (7.1 billion gridpoints).

The vertical axis is the measured Flop rate for the first 50 time steps of the simulation. The horizontal axis is the number of nodes used, with 4 processor cores per node. Over 71% parallel efficiency is realized for strong scaling from 8192 nodes (32,768 cores) to 65,536 nodes (262,144 cores).

In additional to scalability, Nek5000 was measured to achieve 27% of peak performance per core on the Blue Gene/P at 8192 nodes, and 19.3% of peak at 65,536 nodes a peak floating point of 172 TFLOPS.

Summary

The Nek5000 runs on the Juelich BG/P demonstrated excellent scaling (71%) to 262,144 cores and sustained 172 TFLOPS. In addition these runs constituted the largest Nek5000 runs to date, with problem sizes exceeding 7 billion gridpoints. This figure is significant, in that it demonstrates, for the first time, the 64-bit integer global addressing feature of Nek5000's central communication kernel.

References

1. P.F. Fischer, J. Lottes, W.D. Pointer, and A. Siegel, Petascale algorithms for reactor hydrodynamics, J. Phys. Conf. Series (2008).

2. P.F. Fischer, W.D. Pointer, A. Obabko, J. Smith, and H. Childs, Simulation of turbulent diffusion in 217-pin wire-wrapped fast reactor subassemblies, Tech. Report Technical Report ANL-AFCI-267, Argonne National Laboratory, 2009.

3. P.F. Fischer and J.W. Lottes, Hybrid Schwarz-multigrid methods for the spectral element method: Extensions to Navier-Stokes, Domain Decomposition Methods in Science and Engineering Series (R. Kornhuber, R. Hoppe, J. Priaux, O. Pironneau, O. Widlund, and J. Xu, eds.), Springer, Berlin, 2004.

4. H.M. Tufo and P.F. Fischer, Fast paral lel direct solvers for coarse-grid problems, J. Parallel Distrib. Comput. 61 (2001), 151–177

Full Scale Simulation of Coronary Arteries in Presence of Red Blood Cells

Amanda Peters, Harvard University
Simone Melchionna, EPFL and Harvard University
Efthimios Kaxiras, EFPL and Harvard University
Jonas Lätt, EPFL
Massimo Bernaschi, Consiglio Nazionale delle Ricerche
Mauro Bisson, Consiglio Nazionale delle Ricerche
Sauro Succi, Consiglio Nazionale delle Ricerche
and Harvard University

Description of the Code

With the present proposal, we intend to carry out, for the first time, a complete simulation of a coronary arterial system, as reconstructed from Multi-Detector Computed Tomography scans, with the goal of observing the distribution of red blood cells in the whole arterial system. To achieve this goal we will exploit a multi-scale Lattice Boltzmann/Molecular Dynamics (LB/MD) simulation method developed by our group and employed by our multi-physics Software Package, MUPHY. The LB solver exploits the intrinsic parallelism of the mesh-based method to enable solving of the fluid dynamic equations. The fluid is capable of accommodating suspended bodies, such as the highly anisotropic red blood cells, and the coupling between blood plasma and red blood cells is taken into account by the simulator. The evolution of red blood cells is described by a Molecular Dynamics algorithm, highly parallelized and equipped with a dynamic task-transfer algorithm to optimize load balancing.

For the hemodynamic solution, our goal requires a mesh with very high resolution in order to resolve the presence of red blood cells and cell crowding close to the arterial walls. With this, we can compute with high accuracy the endothelial shear stress, the crucial cause of atherogenesis (plaque formation) and the subsequent atherosclerosis. With a resolution corresponding to a mesh spacing of 10 μm we will be able to resolve the structure of red blood cells at high concentration (50% in volume) as found in physiological conditions. This will result in 1.6 x 10^9 voxels and 8 x 10^8 red blood cells. If possible, we will also attempt the much more challenging case corresponding to a mesh spacing of 5 μm, resulting in about 13 x 10^9 voxels and 6.5 x 10^9 red blood cells. These resolutions necessitate the use of a large-scale system such as Jugene.

Accomplished Objectives

Our main accomplishments from our time at the workshop are listed below:

- Successfully scaled to the full 72 rack system.
- Increased to resolution of 5 μm.
- Coupled the Lattice Boltzmann solver with 10 million red blood cells.
- Achieved 64 TeraFlops.

This analysis is performed by increasing the size of the partitions at a fixed problem size, in an effort to analyze the impact of the number of cores on the total simulation time. In Table 1, we show the elapsed time per time-step for the full simulation, as well as for the LB and MD components separately. There are a few points of interest. First, the elapsed time decreases significantly with the number of cores, with a speed-up of 43.5 between the $4,096$ versus $294,912$ processor configurations (see Fig. 1), corresponding to a parallel efficiency in excess of 60 % (see Fig. 2). This result is significant since the average number of mesh points per computational core becomes fairly low (*i.e.,* $\sim 3,300$) on the full configuration of 294,912 cores. These figures are basically the same for the MD and LB sections of the application. Second, we notice that the MD and LB sections remain in a fairly satisfactory balance with each other across the whole range of cores, thereby highlighting the excellent quality of the workload partitioning.

Table 1: Breakdown of the elapsed runtime

Cores	LB	MD	LB+MD
4,096	0.4761	0.04633	.5224
147,456	0.0151	0.419	0.0193
294,912	.0088	.0042	.0130

This simulation that we were able to complete at the Juelich Extreme Scaling workshop contained one-billion fluid nodes that were embedded in a bounding space of one trillion. We were able to couple this fluid solver with the concurrent simulation of ten-million red blood cells.

This achievement is the result of several key pieces of work, namely i) the solution of the formidable graph-partitioning problem prompted by the need of evenly distributing the workload associated with the complex arterial geometry, across the entire 72 rack system; ii) methods to preserve load balancing between the fluid-dynamics and the molecular-dynamics simulation in a complex patient-specific geometry; iii) the innovative modeling techniques required to manage the self-consistent fluid-particle interactions in complex geometries.

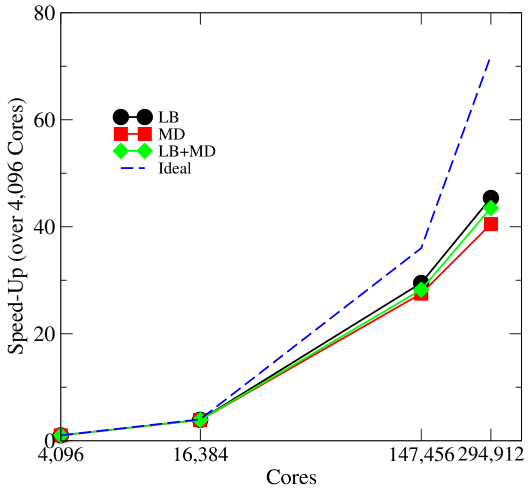

Figure 1: Semilog plot of the speed-up for the LB (circles) and MD (squares) components, for the full simulation (diamonds), and for the ideal regime (dashed line) versus the number of cores.

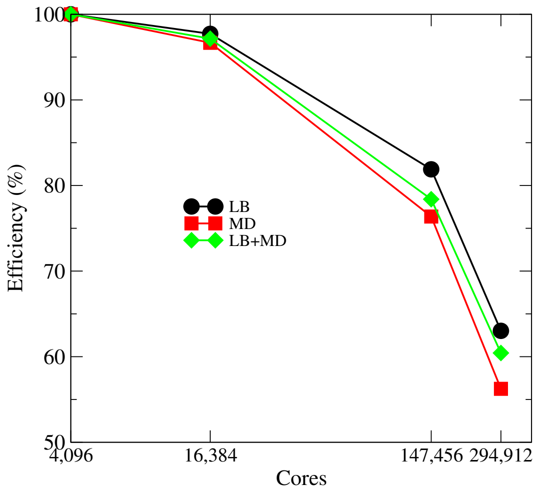

Figure 2: Semilog plot of the parallel efficiency for the LB (circles) and MD (squares) components, for the full simulation (diamonds) versus the number of cores.

Scaling Parallel Fast Fourier Transform on BlueGene/P

Michael Pippig and and Daniel Potts
Chemnitz University of Technology

Description of the Code

The three-dimensional discrete Fourier transform provides the basis of many algorithms in scientific computing and therefore a highly scalable implementation for massively parallel systems such as BlueGene/P is desirable. We consider the three-dimensional input dataset to be of size $N_0 \times N_1 \times N_2$ with $N_0 \geq N_1 \geq N_2$. Unfortunately the FFTW [3] software package, that is appreciated for its portable high-performance FFT implementations, lacks scalability to huge core counts. The reason is that the input dataset is split along one dimension to distribute it on a given number of cores. Hence at most N_1 MPI processes can be used efficiently. Eleftheriou et al. [2] proposed a volumetric domain decomposition to overcome this scalability bottleneck and implemented a software library [1] for power of two FFTs customized to BlueGene/L systems. They split the dataset along two dimensions and therefore were able to increase the number of MPI processes with $N_1 \times N_2$.

During the workshop we tested the scaling behavior of a new parallel complex to complex FFT implementation (PFFT), that combines FFTWs flexible user interface and hardware adaptiveness with the highly scalable two-dimensional data decomposition. Although the parallel FFT implementation from S. Plimpton [5] and the well known P3DFFT [4] already use two-dimensional data decomposition, they miss a large part of FFTWs flexibility. Since we exploit FFTW to implement both ingredients of the three-dimensional FFT algorithm, that are local one-dimensional FFTs and communications in processor groups along the rows and columns of a two-dimensional processor mesh, the implementation can directly transfer FFTWs flexibility to the parallel algorithm. Therefore we are able to generalize our algorithm straight forward to d-dimensional FFTs, $d \geq 3$, real to complex FFTs and even completely in place transformations. Further retained FFTW features like adjustable blocksize, the selection of planning effort via flags and a separate communicator handle distinguish PFFT from other public available parallel FFT implementations. Support of automatic ghost cell creation and truncated FFTs complete PFFTs unique flexibility. It turned out that especially these features are essential to implement an efficient highly scalable generalization of FFT for nonequispaced nodes [6].

Accomplished Objectives

Our main intention was to analyze the strong scaling behavior of FFTs up to the full BlueGene/P machine in Jülich. During the workshop we were able to run FFTs of size 513^3 and 1024^3 on up to 64 of the available 72 racks. Since P3DFFT only supports real to complex FFTs we applied P3DFFT to the real and imaginary part of a complex input array to get comparable times to PFFTs complex to complex FFTs. The test runs consisted of 10 alternately calculations of forward and backward FFTs. Since these two transforms are inverse except for a constant factor, it was easy to check the results after each run. The average wall clock time as well as the average speedup of one forward and backward transformation can be seen in figure 1 for FFT of size 512^3 and in figure 2 for FFT of size 1024^3. Memory restrictions force P3DFFT to utilize at least 32 cores on BlueGene/P to calculate a FFT of size 512^3 and 256 cores to perform a FFT of size 1024. Therefore we chose the associated wall clock times as references for speedup and efficency calculations. Note that PFFT can run these FFTs on half the cores because of less memory consumption. Anyhow we only recorded times on core counts which both algorithms were able to utilize to get comparable results.

Unfortunately the PFFT test run of size 1024^3 on 64 racks died with a RAS event. Nevertheless our measurements show that the scaling behavior of PFFT and P3DFFT are quiet similar. Therefore we expect roughly the same runtime for PFFT of size 1024^3 on 64 racks as we observed for P3DFFT. Our FFT test run of size 1024^3 on 72 racks crashed and exposed a limitation of our data decomposition scheme for very unbalanced data distributions that we were not aware by then.

We also performed separate time measurements of the local FFT calculations and the global communications. This gives an inside to the calculation to communication ratio of FFTs on large core counts as we can see in table 1 for a FFT of size 512^3. Note that 262144 is the maximum number of cores we can efficiently utilize for a FFT of this size. This also means that every core calculates only one local FFT of size 512 in each of the three calculation steps. Therefore the communication takes the largest part of the runtime. The growing communication ratio for increasing core counts also explains the FFT typical decrease of efficiency seen in figure 3.

cores	64	1024	16384	262144
PFFT	53.9%	79.6%	80.5%	96.0%
P3DFFT	59.2%	73.7%	77.2%	95.6%

Table 1: Communication ratio of FFT of size 512^3.

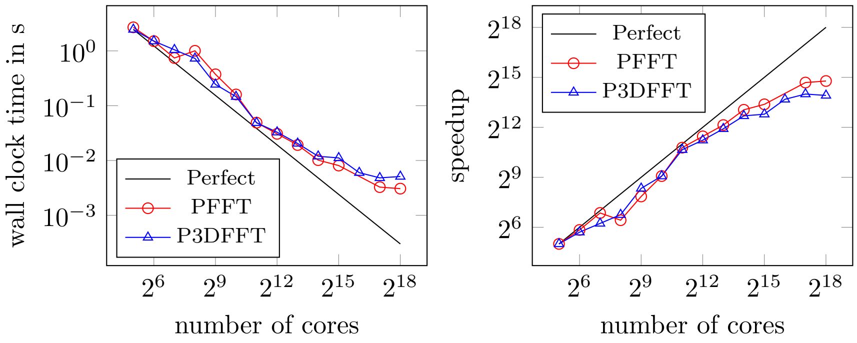

Figure 1: Runtime measurements for FFT of size 512^3.

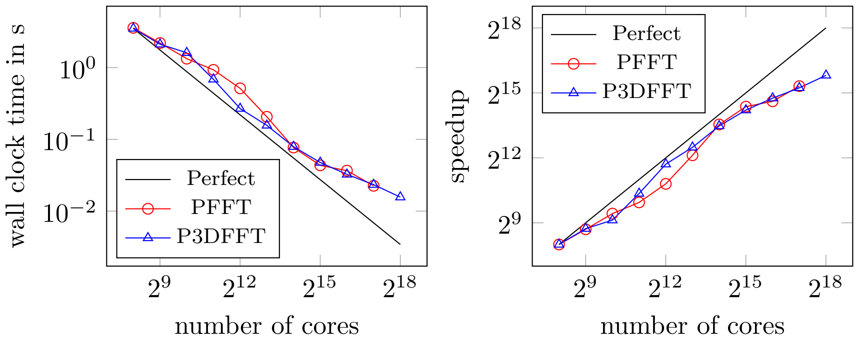

Figure 2: Runtime measurements for FFT of size 1024^3.

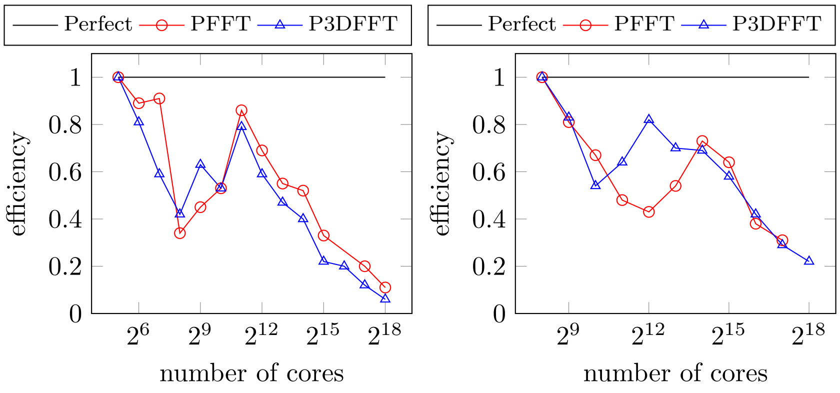

Figure 3: Efficiency for FFT of size 512^3 (left) and 1024^3 (right).

Conclusion

Our runtime tests up to 262144 cores of the BlueGene/P supercomputer prove PFFT to be as fast as the well known P3DFFT [4] software package, while FFTWs flexibility is still preserved. To our knowledge no public available parallel FFT library has been tested to such great core counts by now. These measurements alleviate the decision-making process, whether a parallel FFT should be used to exploit the full BlueGene/P system.

Acknowledgments

We are grateful to the Jülich Supercomputing Center for providing the computational resources on Jülich BlueGene/P. The great support by the workshop organization team improved our handling of the BlueGene/P machine, gave a better understanding of the underlying hardware architecture and even led to new ideas to improve our algorithms. First tests on customized processor mappings in cooperation with Lukas Arnold showed a significant time reduction of the communications with 8192 cores. This encourages us to investigate the impact of customized mappings for greater core counts as well. Conversations with other workshop attendees were very helpful to avoid usage and coding errors in the future. The work of Michael Pippig was supported by the BMBF grant 01IH08001B.

References

[1] M. Eleftheriou, J. E. Moreira, B. G. Fitch, and R. S. Germain. Parallel FFT subroutine library. `http://www.alphaworks.ibm.com/tech/bgl3dfft`

[2] M. Eleftheriou, J. E. Moreira, B. G. Fitch, and R. S. Germain. A volumetric FFT for BlueGene/L. In T. M. Pinkston and V. K. Prasanna, editors, *HiPC*, volume 2913 of *Lecture Notes in Computer Science*, pages 194–203. Springer, 2003.

[3] M. Frigo and S. G. Johnson. The design and implementation of FFTW3. *Proceedings of the* IEEE, 93:216–231, 2005.

[4] D. Pekurovsky. P3DFFT, C subroutine library. `http://www.sdsc.edu/us/resources/p3dfft`

[5] S. Plimpton. Parallel FFT subroutine library. `http://www.sandia.gov/~sjplimp/docs/fft/README.html`

[6] D. Potts. The nonequispaced FFT: An indispensable algorithm for applied science. `http://www.reviews.com/hottopic/hottopic_essay_08.cfm`

Extreme Scaling of the BQCD Benchmark

Hinnerk Stüben
Konrad-Zuse-Zentrum für Informationstechnik Berlin

Momme Allalen
Leibniz Supercomputing Centre, Garching

Description of the Code

We study extreme scaling of the conjugate gradient solver of BQCD *(Berlin Quantum ChromoDynamics program)*. BQCD is used as a benchmark program in procurements at our centres as well as in the DEISA and PRACE projects [1, 2], and it is a code basis in the QPACE project [3].

QCD is the theory of strongly interacting elementary particles. The theory describes particle properties like masses and decay constants from first principles. The starting point of QCD is an infinite-dimensional integral. In order to study the theory on a computer space-time continuum is replaced by a four-dimensional regular finite lattice with (anti-) periodic boundary conditions. After this discretisation, the integral is finite-dimensional but still rather high-dimensional. The high-dimensional integral is solved by Monte-Carlo methods. BQCD is a program that simulates QCD with the Hybrid Monte-Carlo algorithm.

Hybrid Monte-Carlo programs have a compute intensive kernel, which is an iterative solver of a large system of linear equations. In BQCD we use the standard conjugate gradient solver. Depending on the physical parameters 80 % or up to more than 95 % of the execution time is spent in the solver. The dominant operation in the solver is the matrix times vector multiplication. In the context of QCD the matrix involved is called *hopping matrix*. The hopping matrix is large and sparse. The entries in a row are the eight nearest neighbours of one site of the four-dimensional lattice.

QCD programs are parallelised by domain decomposition. The nearest neighbour structure of the hopping matrix implies that the boundary values (surfaces) of the input vector have to be exchanged between neighbouring

lattice: $64^3 \times 128$ (on 72 racks: $64^3 \times 144$)						
#racks	#cores	Mflop/s per core	overall Tflop/s	local lattice	boundary exchange	global sums
1	4.096	341	1.40	$16 \times 8 \times 8 \times 8$	9 %	1 %
2	8.192	335	2.74	$16 \times 8 \times 8 \times 4$	18 %	5 %
4	16.384	324	5.31	$16 \times 8 \times 4 \times 4$	24 %	6 %
8	32.768	333	10.9	$16 \times 4 \times 4 \times 4$	27 %	7 %
16	65.536	301	19.7	$16 \times 4 \times 4 \times 2$	31 %	8 %
32	131.072	250	32.8	$16 \times 4 \times 2 \times 2$	38 %	9 %
72	294.912	203	59.8	$16 \times 2 \times 2 \times 2$	50 %	14 %
lattice: $96^3 \times 192$ (on 72 racks: $96^3 \times 216$)						
#racks	#cores	Mflop/s per core	overall Tflop/s	local lattice	boundary exchange	global sum
1	4.096	380	1.56	$24 \times 12 \times 12 \times 12$	14 %	3 %
32	131.072	366	48.0	$24 \times 6 \times 3 \times 3$	20 %	4 %
72	294.912	353	104.1	$24 \times 3 \times 3 \times 3$	25 %	5 %

Table 1: Strong scaling results for the conjugate gradient solver of BQCD. The last columns indicate the fraction of time spent in communication originating from boundary exchange (hopping matrix multiplication) and global sums (dot products in the solver). (We do not understand why the fractions are relatively high for the large lattice on 1 rack.)

processes in every iteration of the solver. The boundary exchange is communication intensive because the local lattices are typically small. At the single CPU level QCD programs benefit from the fact that the basic operations involve the small complex matrices. However, communication loss can become quite pronounced (see Table 1).

It is interesting to use BQCD as a benchmark because we understand it very well and it is a good indicator for the quality of the communication network of a parallel computer. At an abstract level it represents one class of supercomputer applications: iterative solvers with matrices appearing in problems with nearest neighbour stencils on Cartesian grids.

BQCD is written in Fortran and parallelised with MPI and OpenMP. In real production the hopping matrix multiplication is very often implemented using low level programming techniques [4, 5]. Here we use the Fortran program which is a portable benchmark.

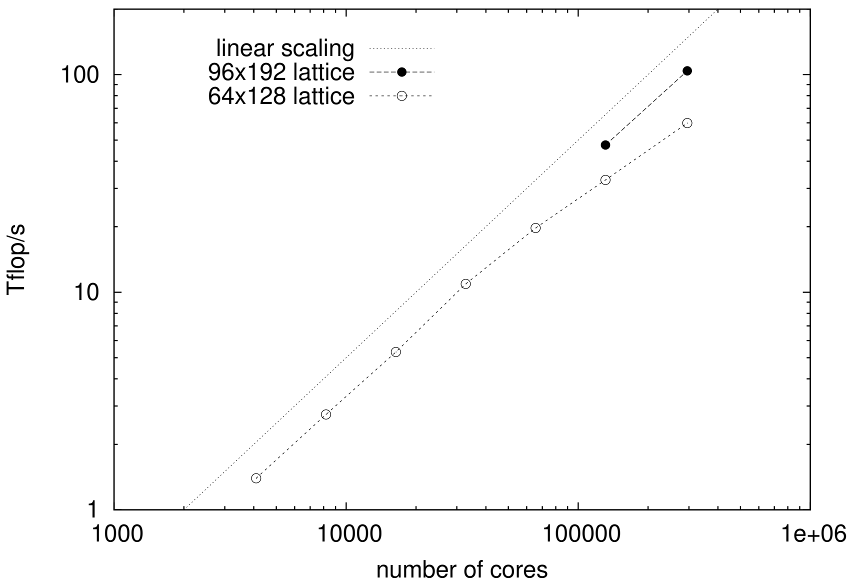

Figure 1: Strong scaling of the conjugate gradient solver of BQCD for $96^3 \times 192$ and $64^3 \times 128$ lattices. The dotted line indicates linear scaling. Any linear scaling is parallel to this line.

Accomplished Objectives

Strong scaling. We studied strong scaling for two lattice sizes: $64^3 \times 128$ and $96^3 \times 192$. On the full machine we extended the lattices to $64^3 \times 144$ and $96^3 \times 216$ in order to fit to the torus of the BlueGene network. We chose these lattice because on the full machine the local lattice sizes are the same as the ones used in a scaling study on the BlueGene/L [4]. On the full machine the local lattices are 16×2^3 and 24×3^3 (the dimensions of the torus network are $4\,\text{cores} \times 32 \times 32 \times 72$). The $64^3 \times 144$ lattice was also used in [5].

Our strong scaling results are given in Table 1 and plotted in Figure 1. We see that on the smaller lattice scaling is almost linear up to 8 racks. Performance per core drops if one or more local lattice dimensions become 2. On the full machine 6 % of the peak performance was obtained. It is interesting to compare with [5] where low level programming techniques were employed and performance in this extreme case (the surface to volume ratio is 3.125) is about three times better.

The larger lattices scales well throughout. In this case we measured 10.4% of the peak performance on the full machine.

Mapping. It is important that the decomposition of the lattice matches the physical torus of the communication network. In test runs we have observed that performance drops up to 25 % if the mapping to the machine is bad. We determine a good mapping by analysing the value of the variable `LOADL_BG_SHAPE` and permuting the letters in the `TXYZ` argument to the `-mapping` parameter of *mpirun* accordingly.

Hybrid parallelisation with MPI and OpenMP. If the communication overhead becomes bigger one might profit from a hybrid parallelisation. BQCD has this capability. However, we found that hybrid runs were always a few percent slower than pure MPI runs.

I/O. It was nice to observe that the approach to I/O which was implemented 10 years ago using MPI-1 worked reasonably even on the full machine. We measured I/O rates between 3 and 5 GByte/s which includes reordering and gathering data as well as calculating checksums on the fly.

Summary

Running BQCD on the full BlueGene/P at JSC was very helpful for benchmarking activities. With regard to benchmarking future supercomputers it is important to know from experience that there are no practical limitations in scaling the program to extreme numbers of cores.

References

[1] `http://www.deisa.eu/science/benchmarking/codes/bqcd`

[2] `http://www.prace-project.eu/documents/public-deliverables/PublicRelease-D7.2.pdf`
`http://www.prace-project.eu/documents/d8-3-2.pdf`

[3] `http://en.wikipedia.org/wiki/QPACE`

[4] T. Streuer and H. Stüben, *Simulations of QCD in the Era of Sustained Tflop/s Computing*, Advances in Parallel Computing 15 (2008) 535–542. `http://www.fz-juelich.de/nic-series/volume38/streuer.pdf`

[5] S. Krieg and T. Lippert, *Tuning Lattice QCD to Petascale on Blue Gene/P*, Forschungszentrum Jülich, IAS Series Vol. 3 (2010) 155–164.

Hydrodynamic Turbulence Induced by Sedimenting Particles

M. Uhlmann
Institute for Hydromechanics, Karlsruhe Institute of Technology

Code description

We are investigating the influence of solid heavy particles upon fluid turbulence. The suspension is dilute, i.e. the solid volume fraction is below one percent, and yet particles can have a significant impact upon the macroscopic flow characteristics. Among many scientifically relevant questions the following are of particular interest for us: how is the settling velocity of particles affected by possible collective and/or turbulence effects? What is the average wake structure of the settling particles? What are the characteristics of particle trajectories (dispersion)?

Computationally the present study is aiming at raising the standard for the direct numerical simulation (DNS) of particulate flow by increasing the system size such that $\mathcal{O}(10^6)$ particles (under dilute conditions) can be accommodated. The specific objective of the simulations during the BG/P Extreme Scaling Workshop was to gather information on the behavior of our numerical strategy and its parallel implementation when executing on a large number of parallel tasks, well beyond usual production runs. The challenge is to scale up despite the inherently implicit nature shared by virtually all numerical algorithms for the incompressible Navier-Stokes equations.

The Navier-Stokes equations for incompressible flow around moving solid particles are solved by means of a finite-difference/immersed-boundary method (Uhlmann, *J. Comput. Phys.*, 209(2):448–476, 2005) i.e. we use a stationary uniform and isotropic Cartesian mesh. A parallel multi-grid solver (relying on three-dimensional Cartesian domain decomposition) is applied to the discrete form of the Poisson equation for pseudo-pressure. Due to the implicit discretization of the viscous terms we also need to solve discrete Helmholtz problems (one for each space direction) at each Runge-Kutta substep; these three-dimensional problems are reduced to one-dimensional sweeps by means of approximate factorization, and the resulting tri-diagonal systems are solved by means of the parallel algorithm of Mattor et al. (*Parallel Computing*, 21:1769–1782, 1995). For this purpose we use three additional one-dimensional Cartesian communicators (one for each space direction). It can be seen that the problem requires almost exclusively next neighbor communication (i.e. exchange of "ghost cell" data), based on the 26 neighboring tasks (considering a cubical local grid, there is data in 6 planes, 12 edges and 8 corners to be exchanged). This means that the predominant point-to-point communication is either between direct "neighbors" (in a three-dimensional torus) or between nodes which can be reached by at most two hops ("diagonal" neighbors). Collective communication takes place at two points in our algorithm (iterative linear system solver – evaluating the residual, and time step computation), both requiring "allreduce" operations. Concerning the particle-related communication, the same communication stencil (next neighbors including "diagonal" neighbors) is required for treating particles overlapping the processor boundaries.

Here we consider the case of particles sedimenting under gravity in initially stagnant fluid in a tri-periodic domain, under the conditions which have been studied experimentally by Parthasarathy and Faeth (*J. Fluid Mech.*, 220:485–514, 1990) at a Reynolds number (based upon particle settling velocity and diameter) of 150.

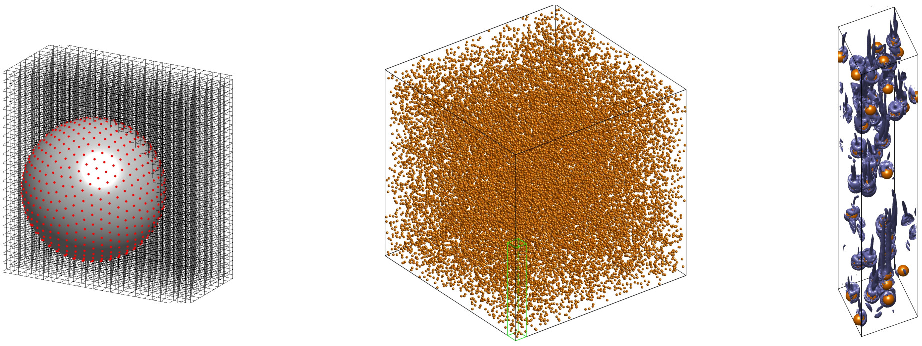

Figure 1: Left: the Eulerian grid around one particle; the red dots indicate the interpolation points on the surface. Center: the instantaneous spatial particle distribution in a preliminary short-time simulation of 27000 sedimenting spheres. Right: corresponding wake structures, depicted in a small sub-volume (green outlined box in center plot). The terminal Reynolds number was 300.

Accomplished Objectives

From the weak scaling results given in table 1 it can be seen that we lose a certain amount of parallel efficiency when increasing the number of processors (not entirely unexpected), but that there is a clear bottleneck when passing from 32 racks (131072 cores) to 64 racks (262144 cores). Incidentally, it can be seen by comparing cases 9a and 9b that the number of particles does not have a significant impact upon the execution time.

In theory, our type of communication is well suited to the network hardware installed on JUGENE. Therefore, we suspect that either (a) in the runs performed so far the MPI tasks were not optimally mapped to the execution cores, or (b) the way the neighbor communication is scheduled in our code (the messages to/from all neighbors are dispatched simultaneously in a non-blocking fashion) was not optimal. Several potential remedies could already be identified during the workshop, and further analysis will be carried out.

In addition, the following issues were verified/solved during the workshop:

- compiler-side optimization, numerical correctness of results,
- I/O and data distribution issues at start-up/termination,
- excluding possible "memory leaks" through open messages,
- adjustment of internal MPI buffer sizes (in particular: DCMF_RECFIFO).

The lessons learned from the workshop will be extremely valuable for the purpose of increasing the parallel efficiency of our application (on BlueGene and other systems), even if typical runs will be limited to 16K-32K cores in the near future.

case	#grid-nodes	#particles	#racks	#cores	exectime/step [s]
6	4096^3	131072	8	32768	119
7	$4096^2 \times 8192$	262144	16	65536	152
8	4096×8192^2	524288	32	131072	195
9a	8192^3	1048576	64	262144	411
9b	8192^3	262144	64	262144	410

Table 1: Timing results ("weak" scaling), recorded for a time step with 15 full multi-grid iteration steps performed over 3 Runge-Kutta sub-steps.

KKRnano: A Program for Large-Scale Density-Functional Calculations

Rudolf Zeller, IAS-3, Forschungszentrum Jülich
Alexander Thiess, GRS and IFF-1, Forschungszentrum Jülich

Description of the Code

In the last two years we have developed a computer program for density-functional electronic-structure calculations for nanosystems with thousands of atoms. This work was motivated by two aims, to avoid the bottleneck of standard density-functional methods, where the computing time increases cubically with the number of atoms, and to make efficient use of massively parallel supercomputers like the Blue Gene/P.

Our program is based on multiple-scattering theory within the Korringa-Kohn-Rostoker (KKR) formulation. It avoids wavefunctions and directly calculates the Kohn-Sham Green function by solving sparse systems of complex linear equations of dimension $N(l+1)^2$. Here N is the number of atoms and l depends on the angular spatial resolution around the atoms, usually $l = 3$ or 4 is sufficient. We solve these equations iteratively by the quasi-minimal residual (QMR) method and calculate the electron density by integrating over an energy mesh with a few tenths of energy points. The main part of the computing time is used in the QMR iterations. This time increases quadratically with N and can be distributed naturally using one processor per atom with good weak scaling efficiency.

Accomplished Objectives

In order to use much more processors than atoms, during the last months we have introduced three additional levels of parallelization by using MPI groups and communicators. One parallelization is over the energy points, which we put into two or there groups to obtain reasonable load balancing despite considerable variation in the number of QMR iterations. The other parallelizations are over the two spin directions in magnetic systems and over the $L = (l+1)^2 = 16$ angular momentum components. The L parallelization is implemented presently only for the QMR iterations, but not in other parts of the program because of the substantial programming work involved.

It was the aim during the workshop to investigate the scaling behaviour of parallelization over energy and L (parallelization over atoms and spin is trivially

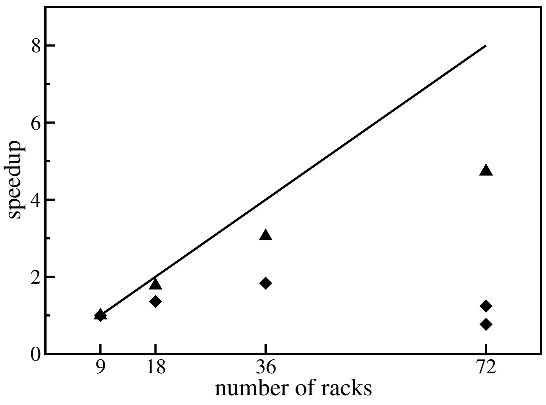

Figure 1: Speedup on Jugene for a system of 3072 atoms using 2, 4, 8 and 16 processors for the parallelization over the 16 L components. Triangles are for the QMR iterations and diamonds for the time occupied by the partition.

rather efficient) and to see whether all 72 racks of the Blue Gene/P could be used for two NiPd test systems with 3072 and 12288 atoms. The smaller system, which required 350 Megabyte per processor, was run in VN mode using pure MPI parallelization, whereas the larger system, which required 1600 Megabyte per compute node, was run in SMP mode using OPENMP with four threads for the L parallelization. For both systems all 72 racks with 294912 processors could be used directly without problems.

For 12288 atoms we considered only the energy parallelization. With 72 racks compared to 24 racks the QMR time was reduced by a factor 2.3, whereas disappointingly the partition time increased by 6 %. For 3072 atoms the speedup obtained by energy parallelization using 9 instead of 3 racks was 2.8 for the QMR time and 2.5 for the partition time. Here the L parallelization was advantageous up to 72 racks for the QMR time, but only up to 36 racks for the partition time (see figure). Disappointingly, the partition time on the full machine was larger than for 9 racks. We used Scalasca to analyse the behaviour using 9 and 18 racks and found considerable time spent after the major calculations. We removed some unnecessary output information (and communication) and obtained a better, but still not satisfactory speedup for 72 racks, marked by the upper diamond in the figure.

In conclusion, we gained confidence that our progam KKRnano is suitable for more than 100000 processors. Scaling results and analysis performed by Scalasca indicate where code changes could be done to improve performance.