John von Neumann Institute for Computing

Jülich Blue Gene/L Scaling Workshop 2006

W. Frings, M.-A. Hermanns, B. Mohr, B. Ort

published in

NIC Symposium 2008,
G. Münster, D. Wolf, M. Kremer (Editors),
John von Neumann Institute for Computing, Jülich,
NIC Series, Vol. **39**, ISBN 978-3-9810843-5-1, pp. 307-314, 2008.

http://www.fz-juelich.de/nic-series/volume39

Jülich Blue Gene/L Scaling Workshop 2006

Wolfgang Frings, Marc-André Hermanns, Bernd Mohr, and Boris Orth

John von Neumann Institute for Computing
Jülich Supercomputing Centre
Forschungszentrum Jülich, 52425 Jülich, Germany
E-mail: {w.frings, m.a.hermanns, b.mohr, b.orth}@fz-juelich.de

From December 5-7, 2006, John von Neumann Institute for Computing, IBM, and the Blue Gene Consortium jointly held the first "Blue Gene/L Scaling Workshop" at Jülich. The aim of this workshop was to provide participants the chance to scale their codes across an 8-rack Blue Gene/L system. Besides the hardware, appropriate software and support personnel were provided to accomplish this task. Jülich provided about 800,000 CPU hours on the 16,384-processor Blue Gene/L system *JUBL* over a three-day period for the scaling runs. The participants of the workshop were selected by a peer review team. Selection criteria were (i) the confidence that the code would scale across 8 racks, (ii) the prospect that the JUBL infrastructure (OS, compilers, libraries) would meet the user's requests, and (iii) the scientific impact that the code could produce. Applicants selected were paired up with assigned advisors from Argonne National Laboratory, IBM, and NIC, who assisted in administrative (log-on, moving data, storing data) and scaling issues. This article highlights some of the results of running, scaling and optimising the participating applications on JUBL.

1 Introduction

In this article we summarise a selection of the results of the first "Blue Gene/L Scaling Workshop" which took place at Forschungszentrum Jülich from December 5-7, 2006. The aim of this workshop was to give both users and the support personnel from Argonne National Lab, IBM, and JSC/NIC the opportunity to investigate the performance and scaling behaviour of selected applications on JUBL, the 8-rack IBM Blue Gene/L system in Jülich.

Unfortunately, the limited space here does not allow for a complete account of all the valuable results and experiences gained during the three days of the workshop. Besides those participants whose results are described below, others have contributed equally to the workshop's big success. Among them were A. Dubey from the University of Chicago, who investigated the performance and scaling of the FLASH code from the field of astrophysics, and A. Nishida from Chuo University, Japan, who studied the scalability of a numerical software library for large scale scientific simulations. The performance analysis of the CFD package XNS with SCALASCA, a tool developed at JSC, proved especially successful for M. Nicolai and M. Probst from the group of M. Behr at RWTH Aachen University. Not only did it result in a significant improvement of the code, it also marked the beginning of a still ongoing, fruitful collaboration between this group and the parallel performance analysis group at JSC.

The following sections describe the outcome of running, scaling and optimising the four remaining participating applications. They are based on the participants' own workshop reports, collected in a JSC Technical Report[1]. For further details, also on the applications omitted here, please consult this reference and the references therein.

2 Molecular Dynamics Studies of Radiation Hard Materials

DL_POLY3 is a classical molecular dynamics package developed at CCLRC Daresbury Laboratory, from where it was brought to the workshop by I. J. Bush. Prior to the workshop, the code had never been run on systems with significantly more than 1000 processors. The system investigated at the workshop was a model of radiation damage in a fluoritized Zirconium pyrochlore. One of the native Gadolinium ions in the system was replaced by Uranium, which was then given a velocity consistent with a 100 keV recoil after an alpha decay. Due to the very high velocity of the Uranium ion it was necessary to study very large supercells, and the total system size used was approximately 14.6 million particles. The power of the Jülich Blue Gene/L allowed, for the first time, to model this system with a realistic recoil.

The DL_POLY3 code is a totally distributed memory code which uses a link-cell algorithm that scales approximately like $O(N)$. The various terms of the force field can be generalised as (i) short range repulsion, (ii) Van Der Waal's (VDW) attraction, and (iii) Coulomb forces.

The first two are both short range terms, and are handled together in DL_POLY3. Subsequent references to VDW terms therefore include both these terms. Due to the short range of these forces they are expected to scale very well with processor count due to their spatial locality (compare halo exchange algorithms). On the other hand, Coulomb forces are long range terms, and have to be handled differently. The Ewald sum technique used in DL_POLY3 splits the evaluation into two terms – one short ranged, one long ranged. While the former is evaluated in real space (very similar to the VDW terms), the long range term has to be handled differently. DL_POLY3 uses the Smooth Particle Mesh Ewald (SPME) algorithm, the key feature of which is a Fast Fourier Transform (FFT). For this DaFT is used, a package written at Daresbury Laboratory that is novel in that it avoids performing 'all-to-all' operations by parallelising the individual 1D FFTs.

Results

Once ported, the code ran and scaled very well "out of the box". The good scaling for MD of the test system is shown in Fig. 1. All jobs were run in virtual node mode.

Of the various elements of the force field the VDW and short range Ewald terms both scale almost perfectly, and at least for VDW terms the expected deviation from ideal behaviour at 16384 processors is not very apparent. As expected, the long range Ewald terms, i.e. those terms that require an FFT, scale less well. However, given the comparatively small size of the FFT grid, $512 \times 512 \times 512$, the scaling is still good.

The absolute times per time step for each of the components of the execution showed that, at these processor counts and for this system, the dominant term was the long range component of the Ewald summation. The most important result, however, was that the time for an MD time step on 16,384 processors was sufficiently small to allow full simulations to be performed in a realistic amount of time.

The one major problem that was experienced was I/O, which was prohibitively slow. This was due to the fact that, at the time of the workshop, all I/O in DL_POLY3 was performed in serial, i.e. all through one processor, and to/from formatted files. The reasons for this were simplicity and portability of the files, and that the time taken for I/O had

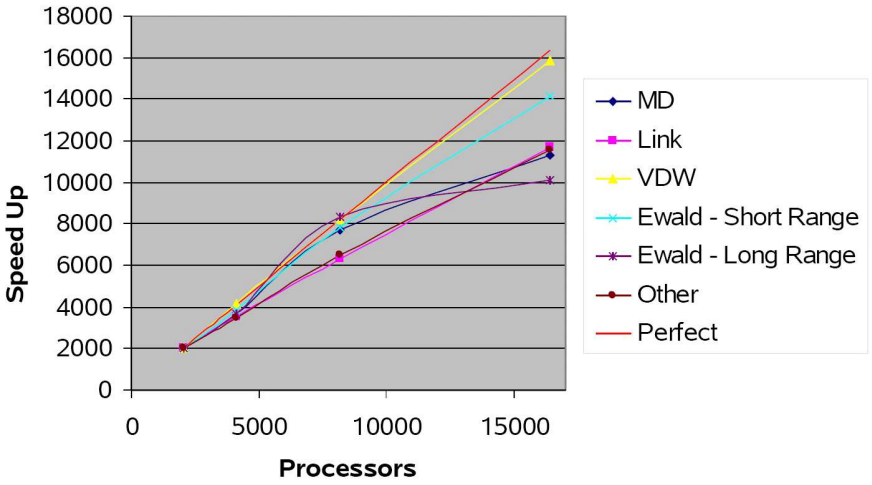

Figure 1. Scaling of DL_POLY3 on JUBL. ”MD” shows the scaling of the total computation time, ”Link” refers to the time taken at each time step to implement the link-cell algorithm (and build the Verlet neighbour list), and the remainder are the components of the force field terms outlined above.

always been small compared to the compute time. However, it became very clear during the workshop that the large system sizes enabled by the massively parallel Blue Gene/L require new, parallel ways of implementing I/O.

Summary

DL_POLY3 was demonstrated to scale well out to 16,384 processors on Blue Gene/L. Porting was straightforward. It was shown that the code runs fast enough on 8 racks to allow a detailed scientific study of the system. Two candidates for optimisation which were found using Xprofiler and explicit timings were the loop nest that interpolates the ionic charges onto the regular grid, and the message passing time in the FFT.

3 Large Scale *Ab Initio* Calculations of Functional Materials

Ab initio Density Functional Theory (DFT) codes have become a widely used tool for the investigation of unusual materials properties and the prediction of novel functional materials. Unlike empirical and semi-empirical methods their accuracy is not hampered by simplifying assumptions about the interatomic interactions, so that their use becomes inevitable where electronic and structural properties become closely interrelated, as it is often the case for modern functional materials. Off-the-shelf DFT codes like VASP (Vienna *Ab initio* Simulation Package), which was investigated at the workshop by M. E. Gruner from the group of P. Entel at the University of Duisburg-Essen, were historically not designed to run large problems on a vast number of processors. However, a complete redevelopment appears infeasible due to their complexity. The readiness of existing codes for modern massively parallel supercomputer architectures is therefore pivotal for the question whether *ab initio* materials science on the mesoscopic scale will become possible in near future.

In previous tests it could be shown that the code scales well up to 1,024 nodes. However, further scaling was hampered by parts of the code which make heavy use of the SCALAPACK eigensolver `pdsyevx` and the parallel 3d Fast Fourier Transforms (FFT). One strategy to circumvent this limitation is the tackling of larger systems, where the relative amount of time spent for interprocess communication will be reduced. The option to optimise the communication structure in the code was not projected at this stage, but may become a target for future efforts. Therefore, three questions were at the centre of interest during the scaling workshop: (i) Is the VASP code capable of running on 8k nodes? (ii) How severe is the limitation of 512 MB per node for larger problems on several thousand nodes? (iii) What is the maximum system size that can be tackled on the Blue Gene/L?

To answer these questions, realistic test systems were simulated. The first system consisted of super-cells of the magnetic shape alloy Ni_2MnGa of various sizes (576, 672, 720, 768, 800, 896 and 1024 atoms), sufficiently large to contain a martensitic twin boundary. These twin boundaries can be shifted in realistic magnetic fields which gives rise to the so-called ferromagnetic shape memory effect, which makes these alloys interesting for a new class of magneto-mechanical actuators. In addition, calculations of large super-cells of the dilute magnetic semiconductor ZnO:Co and GaN:Gd were performed. Three different system sizes were categorised, consisting of a total number of 432, 864 and 1296 atoms. In both cases, due to the large super-cells, integration in k-space was restricted to the Γ-point.

Results

During the workshop it became evident that only 512 MB of memory per node represent a severe limitation. To alleviate this restriction, the coprocessor mode was used throughout all of the calculations.

The largest system that could be simulated on one and two racks was a Ni_2MnGa supercell consisting of 672 atoms (6720 spin-polarised valence electrons). Here, a speedup of 1.73 between one and two racks could be achieved. The largest system that could be tested successfully on the full 8 racks was an 800 atom Ni_2MnGa supercell comprising 8000 spin-polarised valence electrons. Here, a few self-consistency steps could be achieved, which gave enough data for a timing analysis. On a cube (2×2 racks) this problem was brought to complete self-consistency. The measured speedup between four and eight racks, however, was only 1.22. Reasonable timing data could not be obtained for two racks and less due to memory restrictions. However, for a previous test case (561 iron atoms), which was not hampered by memory limitations down to 128 nodes, a speedup of 1.31 between one and two racks was achieved, while on 1024 nodes an efficiency of 70% of the ideal performance, extrapolated from 128 nodes, was measured. It appears plausible to expect an overall efficiency of above 50% for the 800 atom super-cell on the cube. For the doped ZnO the largest system calculated consisted of 864 (7725 valence electrons) atoms which could be computed on one rack.

Summary

From the experience gained at the workshop it was concluded that on a Blue Gene/L system with 512 MB per node the maximum partition that can be used efficiently is four racks.

The largest problems that were manageable with VASP on such an installation contained 800-900 atoms and up to 8000 (spin-polarised) valence electrons. On Blue Gene (L or P) systems with 1 GB per node larger systems can be treated, and efficient scaling will probably be achieved even on 8 or 16 racks. Improvements in the FFT communication scheme, which are planned in close cooperation with IBM, might result in significant performance gains.

4 Turbulent Convection with Very Large Aspect Ratios

Turbulence often occurs in geometries where the lateral dimensions exceed the vertical dimension by orders of magnitude. For example, atmospheric mesoscale layers typically have lateral extensions of up to 1000 km and are characterised by aspect ratios $L : W : H = 1000 : 1000 : 1$.

At the workshop, J. Schumacher from the Technical University of Ilmenau studied turbulent Rayleigh-Bénard convection, which represents one particular process in such mesoscale layers. His three-dimensional pseudospectral simulation code advances the Boussinesq equations for an incompressible fluid in time by means of a second-order predictor-corrector scheme. Lateral boundary conditions are periodic; vertical boundary conditions are free-slip.

One of the core parts of the numerical scheme used in this simulation is the Fast Fourier Transform (FFT). The classical parallel implementation of three-dimensional FFTs uses a slab-wise decomposition of the simulation domain. For a simulation with N^3 grid points, the method allows a parallelisation on up to N processors. Due to the rather small memory size per core, the Blue Gene/L requires a *volumetric* FFT which decomposes the three-dimensional volume into cuboid-like rods and hence allows a parallelisation degree of N^2.

The prime requirement for a simulation with a large grid is that the subdomains of the grid (including buffers and temporary storage) fit into the 512 MB of memory on a single Blue Gene/L node. At the same time the FFT algorithm should of course be scalable, i.e. increasing the number of CPUs to solve the problem should also substantially decrease the time-to-answer.

Results

Three FFT packages were compared during the workshop: the old slab-wise method, the BGL3DFFT by M. Eleftheriou *et al.*, and the P3DFFT package by D. Pekurovsky. After some tests the P3DFFT package turned out to be the best solution in terms of performance. Moreover, its interface and implementation met the application's needs in an optimal way (real-to-complex/complex-to-real).

In addition to the inclusion of the P3DFFT package, further optimisations contributed to the improvement of the code performance. These include the optimisation of the process topology, loop-splittings to enable code vectorization in the calculation of the r.h.s. of the Boussinesq equations, and the use of non-blocking point-to-point communication instead of MPI_Reduce and MPI_Bcast.

Further analysis indicated that with these improvements almost all of the remaining communication overhead is related to calls to MPI_Alltoallv required by the FFT algorithm. They are thus the only limiting factor for strong scaling. Future improvements

CPUs	processor mode	$iproc \times jproc$	time
1024	co	32×32	205.7 s
2048	vn	32×64	183.3 s
4096	co	64×64	90.66 s
4096	vn	128×32	109.6 s
4096	vn	64×64	118.6 s
8192	co	128×64	60.5 s
8192	co	512×16	74.0 s
8192	vn	128×64	77.8 s
16384	vn	128×128	62.6 s

Table 1. Runtime tests for the aspect ratio $\Gamma = 32$. The CPU time was measured with MPI_Wtime() for a loop of 5 time steps.

of the implemenation of MPI_Alltoallv will therefore have an immediate impact on the performance of this application.

Table 4 summarizes the findings for a grid resolution intended for production runs: $N_x \times N_y \times N_z = 4096 \times 4096 \times 256$ at $\Gamma = 32$. For such a large grid it turned out to be necessary to implement some internal summation loops in double precision.

The parameters $iproc$ and $jproc$ in the table represent the two dimensions of the processor grid that has to be generated for the volumetric FFTs. The differences in execution time for coprocessor or virtual node mode are small for large problem sizes, which is due to the fact the local subdomains no longer fit into the L3 cache of a Blue Gene/L node, so that the two cores compete for the memory bandwidth.

Summary

The use of the P3DFFT package and a number of further improvements in the communication overhead improved significantly the performance of this code on Blue Gene/L. These efforts allowed to run production jobs on at least two racks. Since the code performance is bound by the memory bandwidth for problem sizes of interest, VN mode does not result in a significant speedup compared to CO mode.

5 Numerical Simulations of QCD

The objective of current projects of the QCDSF collaboration is to better constrain the extrapolation of lattice results to the chiral and continuum limits by performing simulations at more realistic quark masses and at smaller lattice spacings. This has become possible due to substantial improvements in the Hybrid Monte Carlo (HMC) algorithm and a significant increase in computing power.

The program investigated by H. Stüben from ZIB Berlin is called BQCD (Berlin QCD). It is a Hybrid Monte Carlo code that simulates quantum chromodynamics (QCD) with Wilson gauge action and non-perturbatively $O(a)$ improved Wilson fermions. The program is written in Fortran 90 and uses MPI. For Blue Gene/L the most compute intensive part has been implemented in assembler.

Lattice QCD is defined on a four-dimensional regular lattice with (anti-)periodic boundary conditions. The kernel of BQCD is a standard conjugate gradient solver with even/odd preconditioning. Typically, 80 % of the execution time is spent in this solver. The dominant operation is the matrix-vector multiplication. In the context of QCD the matrix is called *fermion matrix*. This fermion matrix is sparse with eight entries per row. The entries in row i are the nearest neighbours of entry i of the vector.

The entries of the fermion matrix are 3×3 complex matrices and the entries of the vector are 3×4 complex matrices. Experience on current machines shows that the performance of the Fortran code for the matrix-vector multiplication is about 10 % of peak, and approximately 20 % in case of the assembler version. On the Hitachi SR8000-F1, which was one of the main production machine for the QCDSF collaboration from 2000–2006, the Fortran code ran at about 40 % of peak.

Parallelism of the program is achieved using domain decomposition. In the parallelised *cg*-kernel ghost cells of the input vector have to be exchanged before the vector can be multiplied with the fermion matrix. In order to scale QCD programmes up to high numbers of processes an excellent communication network is required because the lattice volume per core becomes small. On the other hand small lattice volumes per core improve the utilisation of data caches and hence improve performance.

For the floating point operations of the fermion matrix multiplication the SIMD *(double hummer)* instructions of the floating point units are used, which can perform four floating point operations per clock cycle. The communication is done in parallel with the computation, so that in principle computation and communication can overlap. For the communication between nearest neighbours in three of the four directions the torus network is used. The network is accessed from the nodes via memory-mapped fifos. Because of the simple communication pattern (of nearest neighbour data exchange only), no dynamical routing is necessary. Furthermore, each node receives the data packets in the order in which they are needed, so that no reordering of packets is necessary. Each node sends a packet to its neighbour one iteration before the neighbouring node needs the data, so that the network latency can be hidden.

In the fourth direction, which corresponds to the x-direction of the lattice, the lattice is split between the two CPUs in one node, so that the communication can be done via the shared memory. However, since the L1 caches are not coherent, the straightforward approach of simply reading data from the other CPU's memory does not work. At present, a memory region in the L3 cache (the scratchpad area) is used, which is marked L1-caching inhibited for data exchange between the two cores.

Results

In scaling tests during the workshop the performance of the *cg*-kernel was measured. To this end the code was instrumented with timer calls, and for the kernel all floating-point operations were counted manually.

In order to get good performance it is important that the lattice fits the physical torus of the machine. In order to achieve this, MPI process ranks have to be assigned properly. On Blue Gene/L this can be accomplished by setting the environment variable `BGLMPI_MAPPING` appropriately.

The results of the performance tests are illustrated by double logarithmic plots in Fig. 2 (the dotted lines indicate linear scaling). As can be seen from the plots the Fortran/MPI

version exposes super-linear scaling on the $48^3 \times 96$ lattice. Even the $32^3 \times 64$ lattice scales quite well given the fact that the lattice volumes per core are tiny. For the same tiny local lattices the scaling of the assembler version is even better than that of the Fortran/MPI version. This means that in the assembler version computation and communication really overlap.

5.1 Summary

At the workshop it was found that the BQCD code scales up to the full 8 racks of the Jülich Blue Gene/L. The highest performance measured was 8.05 TFlop/s on the whole machine. This high performance could be obtained by using *double hummer* instructions and techniques to overlap communication and computation.

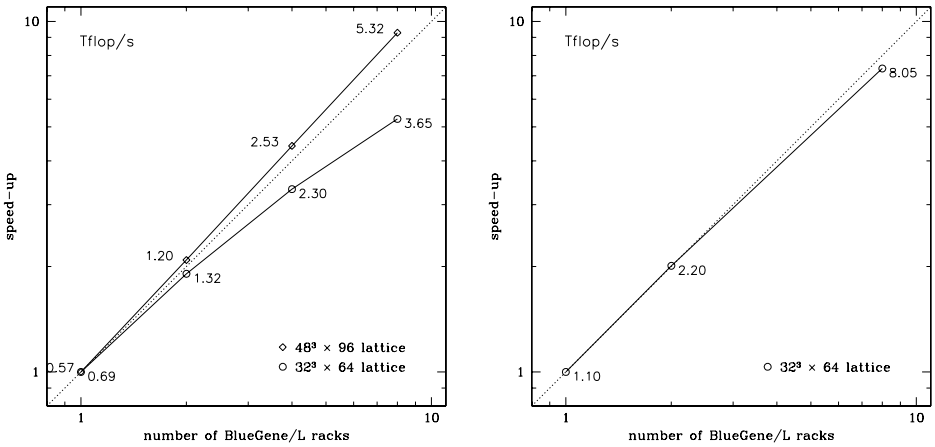

Figure 2. Scaling of the *cg*-kernel of BQCD in the Fortran 90/MPI version (left), and in the assembler version (right).

Acknowledgments

The authors would like to thank their co-organizers from IBM and the Blue Gene Consortium, all participants, and last but not least all their colleagues from JSC who helped making this workshop a success.

References

1. W. Frings, M.-A. Hermanns, B. Mohr, B. Orth (Eds.) *Blue Gene/L Scaling Workshop 2006*, Technical Report IB-2007-02, 2007. For an online version see `www.fz-juelich.de/jsc/files/docs/ib/ib-07/ib-2007-02.pdf`.