PANDORA Upgrade: Particle Dispersion in Bigger Turbulent Boxes

Thorsten Wittemeier, University of Southampton

David Scott, Edinburgh Parallel Computing Centre

John Shrimpton, University of Southampton

Version 1.2, March 22, 2019

1. Introduction

PANDORA [6, 8] is a pseudo-spectral flow solver providing solutions to the Navier-Stokes equations in a $2\pi^3$ domain for isotropic and homogenous turbulent flow using FFTW [5] and MPI. It also includes the tracking of a large number of point Lagrangian particles. PANDORA is a pseudo-spectral direct numerical simulation code and as such does not need any halos for parallelisation. However, the Lagrangian particles stay within the part of the fluid domain associated to the process they are on. For computing the equation of motion for the particles it is necessary to interpolate the fluid velocity at the position of the particle. In particular when higher order interpolation schemes are used, fluid information from parts of the domain associated with other processes is needed. This information is provided by halos. The performance of PANDORA, in particular scalability and memory use, is heavily dominated by the way the halos are implemented.

A new version of PANDORA based on the libraries FFTW and PETSc [1, 2, 3] was developed within this eCSE project. The aim of the project was to overcome the limitations of the previous code in terms of memory use and efficiency. The parallelisation was changed from a 1D ('slab') decomposition to a 2D ('pencil') decomposition. All of the parallelisation was implemented using distributed arrays as provided by PETSc. The global transforms that are necessary to perform FFT in all three directions have been implemented using the parallel transpose functions of FFTW.

1.1. Performance issues with the previous code

Several issues prevented the use of the code for large-scale computations. First of all, a significant amount of the data stored in the old code is not necessary. In a pseudo-spectral DNS code, real-space information is only needed for computing the non-linear term of the Navier-Stokes equation from the velocity and vorticity (in the rotational formulation). When particles are included, an additional use of real-space information is the interpolation of fluid velocities at the particle position. It is therefore not necessary to permanently store the real-space information. Once the non-linear term is computed, the vorticity is also not needed any further. The 1D spatial decomposition (slab decomposi-

tion) determines the memory load per compute node and a puts a limit to the maximum size of simulations domains. The use of halo layers compounds this issue. There is a MPI overhead with this method - splitting the sub- domains into thinner and thinner slices (ie more cores per job) increases the parallel data transfer of the fluid information and the proportion of halos in the memory. Furthermore the parallelisation of the old code was inflexible and required 2^n number of cores, which means on ARCHER nodes typically 8 out of 24 cores are idle. The size of the velocity halo can be drastically reduced by simply changing the 1D (slab) decomposition to a 2D (pencil) decomposition.

Dealiasing in PANDORA is performed using the 2/3 rule, which means that the 1/3 highest wavenumbers are set to zero. Taking into account all three dimensions, this means that only $(2/3)^3 = 8/27$ of the arrays contain non-zero values. The Fourier transforms are performed in 1D and 2D in the slab decomposition and in 1D in the pencil decomposition. Therefore it can be avoided to store zeroes by zero-padding of the buffered array before each transform. The particle arrays are allocated to a fixed size which accounts for a possible higher number of particles on a given process. By using dynamic allocation, a further reduction in memory use can be achieved.

Two fluid arrays (one in real space, the other in wave space) were identified that could easily be replaced by scalar variables within the loop for the computation of the Navier-Stokes equation, thus freeing up a significant amount of memory.

In PANDORA, the real space arrays are used to store coordinates, fluid velocity, fluid vorticity, Runge-Kutta buffer and the non-linear term. All of these arrays include a halo, whereas this is only (temporarily) necessary for the velocity and only for the interpolation of the fluid velocity at the particle position. The coordinates do not need to be stored at all. They are sufficiently simple to compute them directly when needed. Similarly, the wave number does not need to be saved and can be computed when needed.

1.2. Goals of the eCSE project

The first goal of the project was to reduce the per-node memory usage. The minimum success metric was to undertake computations on ARCHER of at least 4096^3 grid points on 128 ARCHER nodes and ideally 8192^3 domains on around 1000 ARCHER nodes.

This involved changing the domain decomposition from 1D to 2D as well as using the real-space data for computing the non-linear terms and solving the Maxey-Riley equation for the Lagrangian particles immediately after the Fourier transform rather than storing the real-space data permanently. A further part of this work package was to use the halo arrays only for the fluid velocity and only when PANDORA is being used with particles.

Another main objective was to improve the parallel I/O performance of the code and speed up the data transfer as well as enable the use of HDF5 and NetCDF. First tests of parallel particle restart files had already been successfully performed using MPI I/O prior to the eCSE project. However, the unequal number of particles per process leads to additional complications not present for the fluid restart files. An additional part of the project was to open-source and document the code and to demonstrate scaling up to 1000 cores.

2. Implementation of the new PANDORA 2.0 code

The PANDORA code was reimplemented in its entirety using parts of the existing code and making extensive use of PETSc. Table 1 summarises the major differences. First of all, the 1D decomposition of the old code was replaced with a 2D decomposition. This was achieved using distributed arrays as provided by the PETSc library. Whereas the 2/3 rule is used for dealiasing in both codes, the higher, i.e. aliased, wave modes are not stored in PANDORA 2.0, thus saving a significant amount of memory. Only those arrays that are strictly necessary are stored in the new implementation of PANDORA.

Most of the program flow is executed on 2D slabs, the orientation of which is dictated by the consecutive Fourier transforms and global transposes. In order to achieve this, we split MPI communicators using the minimum coordinate in the third direction as colour. This is easily done using PETSc and was implemented in a similar way to that used in the TPLS code, which was optimised in a previous eCSE project [4]. The global transpose between Fourier transforms, which was previously implemented by bespoke MPI routines, has now been delegated to the FFTW-MPI library, which chooses the fastest algorithm at initialisation, as is also the case for the initialisation of the Fast

Fourier Transforms. Code for both in-place and out-of place communication has been implemented, the latter enabling MPI AllToAll communication, but evidently using more memory.

Parallel file input and output uses the MPI-IO routines as defined in the MPI 3.0 standard in both codes. However the new code accesses these routines indirectly through PETSc. This allows for an easy access to other file formats like HDF5 or VTK without major changes. We chose MPI-IO for reading and writing the restart files due to its performance. Using PETSc, small file conversion routines can be easily implemented. As the particle arrays in the old PANDORA code were not implemented in natural Fortran ordering, efficient particle restart file reading and writing in parallel was difficult to achieve. We have fixed this problem in the new code. The number of particles on each process is now updated at each time step and the particles are stored in appropriately sized arrays, whereas the old code had predefined arrays of a fixed size. This led to a significant reduction in memory use.

PANDORA 2.0 makes extensive use of Fortran 2003/2008 functionality where appropriate. In particular the iso C bindings provide a useful means of standardising all variable types. Although the code can be used with the standard 32-bit version of PETSc, it is 64-bit enabled, allowing domain sizes with more than $2\,147\,483\,648 = 2^{31}$ grid points.

The program flow has been slightly modified, mainly to make better use of data already available in memory, as shown in figures 1 and table 2 for the previous code and figure 2 and table 3 for the new code. Both codes use the same memory-saving third-order Runge Kutta time-stepping scheme [10], but as a comparison of tables 2 and 3 shows, it has been reorganised to directly use information in fast memory rather than separating the solution of the Navier-Stokes and Maxey-Riley equations and the Runge-Kutta scheme as in the old code. The equation of motion for the particles is now solved before the Navier-Stokes equation. This has two advantages. First of all, the real-space fluid velocity field can be used directly after computing the non-linear term and can then be discarded. The halos are made accessible for the velocity interpolation at the particle position by obtaining PETSc local vectors from the global vectors (i.e. arrays containing halos as opposed to arrays without). The other advantage of solving the particle equation first is that this

Table 1: Comparison of PANDORA code and PANDORA 2.0

	PANDORA	**PANDORA 2.0**
Decomposition	Slab (1D) own MPI routines	Pencil (2D) PETSc
Number of MPI processes	2^n, no more than number of grid points in the parallelised direction	any; no more than product of the number of wavemodes in the parallelised direction
Halos	all real-space arrays	only velocity (particle simulations)
Dealiasing	2/3 Rule: Set wavemodes $\|k\| > 1/3N$ to zero $\rightarrow 1^3 = 1$	2/3 Rule: Do not save wavemodes $\|k\| > 1/3N$ $\rightarrow (2/3)^3 = 8/27$
Wave-space arrays	Velocity, vorticity, Runge-Kutta, wave numbers	Velocity, Runge-Kutta
1D real-space / 2D wave-space arrays	-	Velocity, vorticity
Real-space arrays	Velocity, vorticity, coordinates	**2D** velocity and vorticity, Particle simulations only: (3D) velocity
Global transforms	own MPI routines	FFTW-MPI
Particles	Maxey-Riley equation (drag and gravity)	Passive Particles and Maxey-Riley equation (drag and gravity)
Velocity interpolation	Polynomial interpolation and Lagrange interpolation (direct 1solution), 1st - 4th order	Lagrange interpolation (Neville algorithm), order only limited by the halo nodes (cannot exceed size of neighbouring processes)
Parallel file I/O	MPI-IO, own routines	PETSc: MPI-IO, but easy to implement other formats like VTK, HDF5
Language	Fortran 90/95	Fortran 2003/2008
Data types	Generic Fortran	Almost exclusively ANSI C: iso_c_bindings, PETSc, FFTW
Documentation	comments in source code only	Doxygen

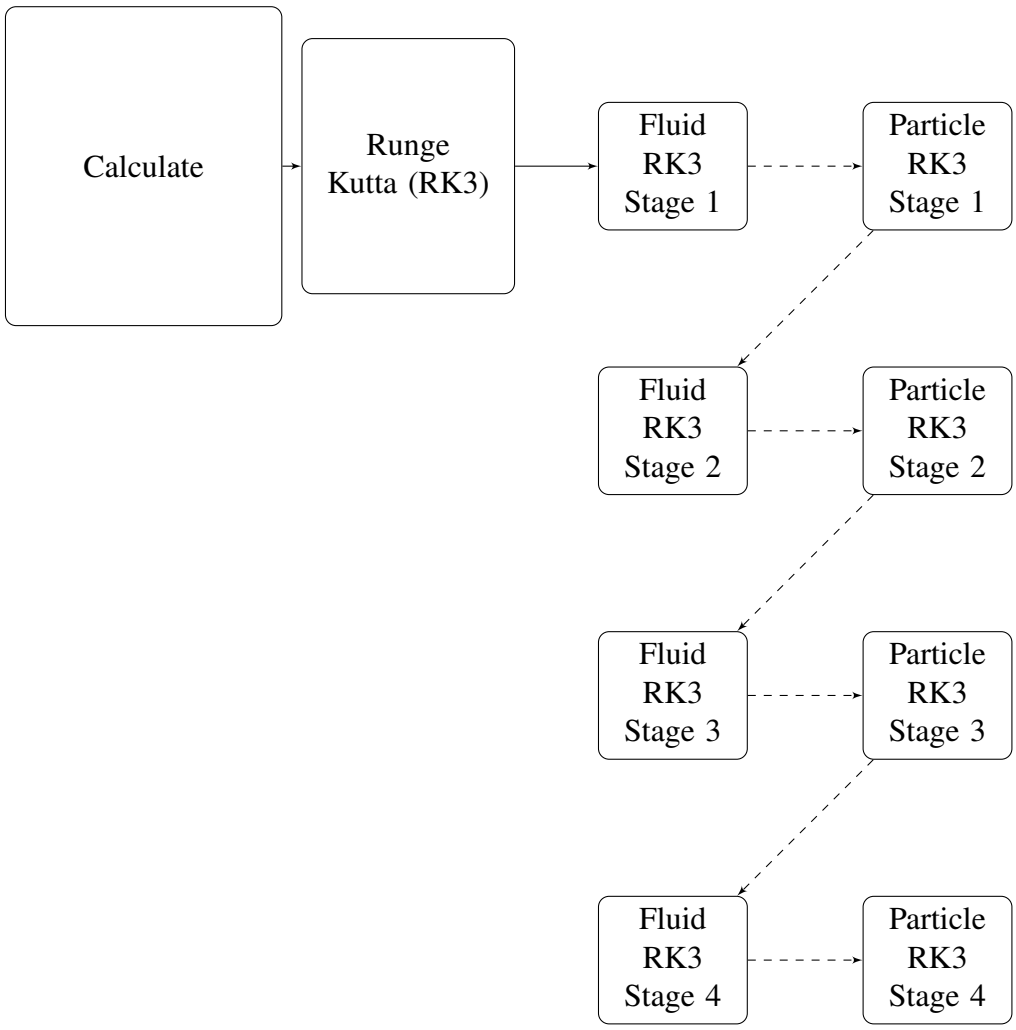

Figure 1: Flowchart of a timestep in PANDORA. In each timestep a Runge-Kutta scheme is performed, followed by fluid and particle analysis. The Runga-Kutta scheme consists of four stages, each of which is composed of a fluid calculation followed by a particle calculation [11].

Stage 1:	$U_1 = U_n$
	$G_1 = F(U_n, t_n)$
Stage 2:	$U_2 = U_1 + \frac{1}{3}\Delta t G_1$
	$G_2 = -\frac{5}{9}G_1 + F(U_2, t_n + \frac{1}{3}\Delta t)$
Stage 3:	$U_3 = U_2 + \frac{15}{16}\Delta t G_1$
	$G_3 = -\frac{153}{128}G_2 + F(U_3, t_n + \frac{3}{4}\Delta t)$
Stage 4:	$U_{n+1} = U_2 + \frac{8}{15}G_3$

Table 2: Runge-Kutta scheme used in PANDORA [11]

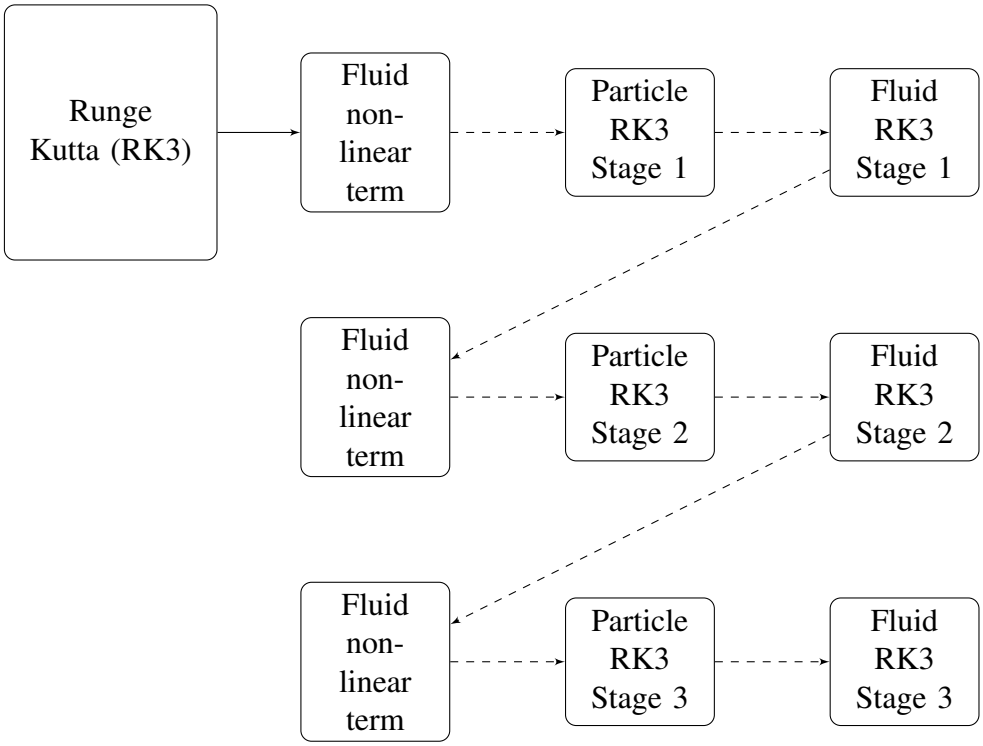

Figure 2: Flowchart of a timestep in PANDORA 2.0. In each timestep a Runge-Kutta scheme is performed, followed by fluid and particle analysis. The Runga-Kutta scheme consists of three stages. As opposed to the previous PANDORA code, the particle Runge-Kutta routines are performed after calculating the non-linear term and before solving the Navier-Stokes equation.

Stage 1:	$U_1 = U_n$
	$G_1 = F(U_n, t_n)$
	$U_2 = U_1 + \frac{1}{3}\Delta t G_1$
Stage 2:	$G_2 = -\frac{5}{9}G_1 + F(U_2, t_n + \frac{1}{3}\Delta t)$
	$U_3 = U_2 + \frac{15}{16}\Delta t G_1$
Stage 3:	$G_3 = -\frac{153}{128}G_2 + F(U_3, t_n + \frac{3}{4}\Delta t)$
	$U_{n+1} = U_2 + \frac{8}{15}G_3$

Table 3: Runge-Kutta scheme used in PANDORA 2.0. The scheme is essentially the same as in the old PANDORA code, but organised differently to directly use information that is already in memory.

allows for a natural way to provide the coupling force in two-way coupled simulations. The particle code has now been modified to allow for passive particles as well as inertial particles. Simultaneous simulations of passive and inertial particles are made available through the particle diameter, where negative diameters are reserved for passive particles (which obviously do not have a diameter or mass).

The main goal of the project was to reduce the memory use of PANDORA to allow for bigger simulations with more particles. This was achieved by storing only the data strictly necessary. Splitting MPI communicators in each of the two parallelised directions makes it possible to perform the main work on Fourier transforms and global transpose on two-dimensional slices in parallel. Therefore less memory is needed for buffer arrays. Since the 2/3 rule for dealiasing is used in PANDORA, it was attempted to avoid real-space data wherever possible. Applying the 2/3 rule in all three dimensions it can be seen that in wave-space only $(2^3)/(3^3) = 8/27$ of memory is needed compared to the corresponding real-space data.

3. Results

3.1. Reduction of memory use

A combination of the above principles led to a significant memory improvement compared to the previous PANDORA code[1] as summarised in table 4. It can be seen that PANDORA 2.0 uses only about a 10th of memory for fluid simulations compared to the previous version.

For simulations with particles the target was not entirely achieved. This is due to the fact that, as opposed to the original planning, it was decided to use three-dimensional real-space velocity arrays for the interpolation of fluid velocities at the particle position. The original idea to use ghost particles rather than fluid halos turned out to be to difficult to implement. It also needs to be emphasised that for many particles (more than one per

[1]Memory use in PANDORA 2.0 is measured exactly using PETSc. For the old PANDORA code, memory use is estimated based on the size of the main arrays. As can be seen from table 4, the memory demands of this code would already go beyond available memory on ARCHER on 2048^3 domains.

Table 4: Memory improvements in PANDORA 2.0

Grid size (real space)	1024^3	2048^3	4096^3	2048^3
ARCHER nodes	32	64	128	64
Cores	512	1024	2048	1536
Particles	-	-	-	8,589,934,592
Memory/node in PANDORA (estimate)	20 GB	80 GB	-	-
Memory/node in PANDORA 2.0 (target)	3.5 GB	14 GB	54 GB	25 GB
Memory/node in PANDORA 2.0 (achieved)	1.9 GB	6.3 GB	23.8 GB	32.5 GB

control volume) fluid halos are more memory saving.

3.2. Enabling simulations on bigger domains

One of the main objectives of this project was to overcome the limitations of the previous code as to the size of the simulation domain. The new code was successfully tested up to 8192^3 domains, using 12288 ARCHER cores. The biggest tests that were performed during the eCSE project are listed in table 5. Although the largest simulation with particles was performed on a 2048^3 grid with one particle per control volume, this is by no means the absolute limit. This grid size is already sufficiently big to test the correct implementation of 64-bit integers for the array indexing, which is a prerequisite for any computing grid or number of particles that goes beyond $2\,147\,483\,648 = 2^{31}$, as is the case for 2048^3 domains. All of these simulations demonstrate that the new code is appropriate for simulations at very competitive sizes with a moderate use of computing resources.

3.3. Performance of fluid simulations

We performed simulations of a relatively small size to compare the old and new codes. The 128^3 was chosen to allow serial simulations with the old, more memory-consuming code. The strong scaling is shown in figure 3 with the speed-up on the left-hand side and the computing time for 10 time steps on the right-hand side. It is immediately ob-

Table 5: Overview of the largest simulations performed so far

Grid size (real space)	4096^3	4096^3	8192^3	2048^3
ARCHER nodes (goal)	128	128	1280	64
ARCHER nodes (achieved)	64	128	512	64
Cores	1536	2048	12288	1536
Particles	-	-	-	8,589,934,592
Memory/node in PANDORA 2.0	46.9 GB	23.8 GB	48.0 GB	32.5 GB

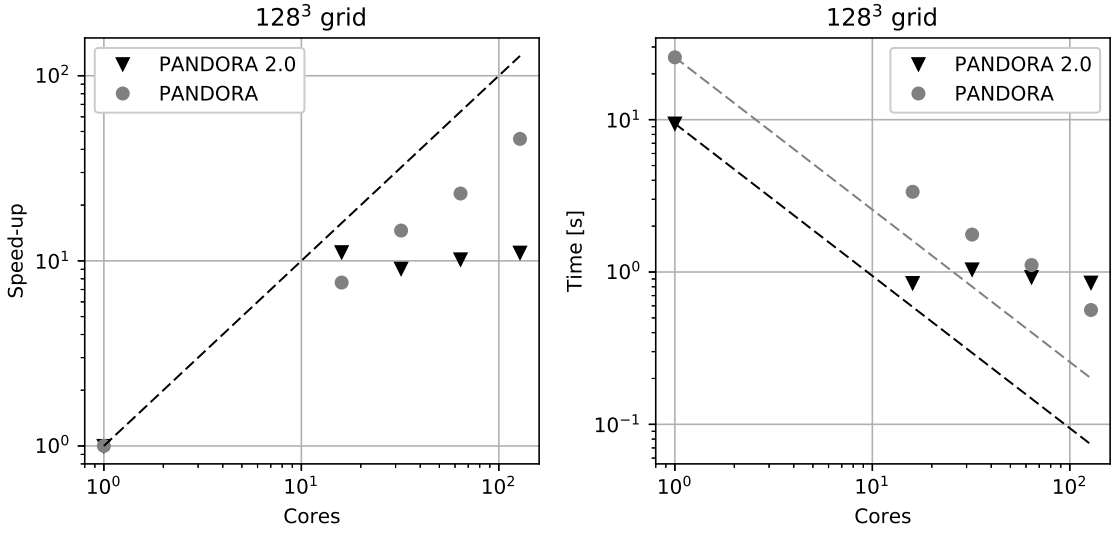

Figure 3: Strong scaling of fluid simulations on ARCHER using PANDORA and PANDORA 2.0. The left-hand side shows the speed-up, the right-hand side the computing time for 10 time steps on a 128^3 domain. The dashed lines indicate ideal scaling.

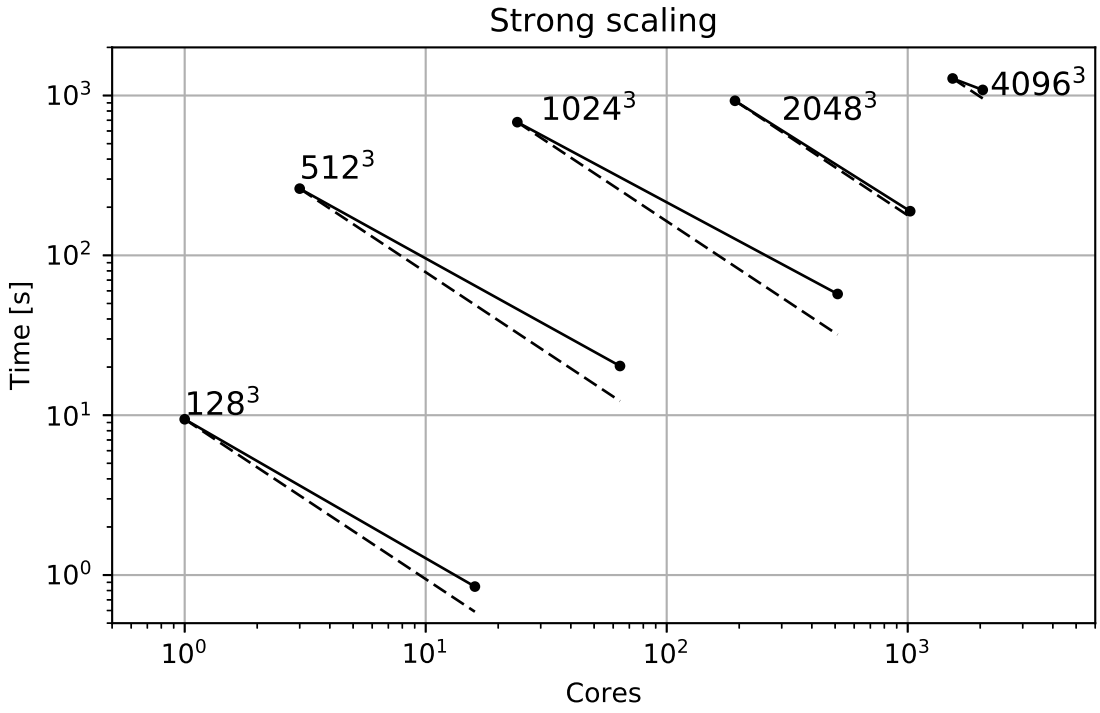

Figure 4: Strong scaling of fluid simulations on ARCHER using PANDORA 2.0. The computing time for 10 time steps is shown for 128^3, 512^3, 1024^3, 2048^3 and 4096^3 domains. The dashed lines indicate ideal scaling.

vious that the scaling behaviour of the new code is more complex. While on 16 cores an almost ideal speed-up is achieved, better than the old PANDORA code, the speed-up stagnates on more cores. The old code on the other hand shows a linear, but less than ideal, scaling behaviour. However in all simulations except for the one on 128 cores the new code is faster. These results can be explained with a significant reduction in computing time as only non-zero values are included in the wave-space arrays, which leads to a communication overhead when more processes than necessary are chosen.

As can be seen from figure 4, the good speed-up at the lower end of the scaling plot can be replicated at larger domain sizes. Although the code does not reach ideal scalability, it does come close when used on a reasonable number of cores.

The weak scaling is shown in figure 5. The left-hand side shows a comparison of PANDORA and PANDORA 2.0 under identical conditions, while the right-hand side

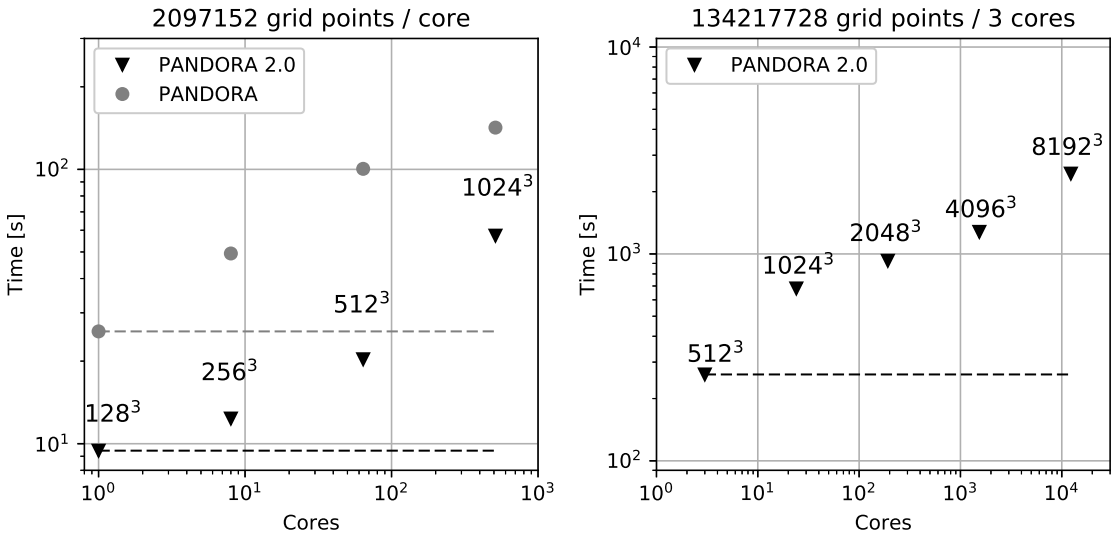

Figure 5: Weak scaling of fluid simulations on ARCHER using PANDORA and PANDORA 2.0. The left-hand side shows the computing time for 10 time steps on 128^3, 256^3, 512^3 and 1024^3 domains, the right-hand side shows the computing time for 10 time steps on 512^3, 1024^3, 2048^3, 4096^3 and 8192^3 domains.

shows larger simulation. It can be seen that, not unexpected for a global spectral code, the communication overhead leads to longer simulation times with growing domain sizes. It is also striking that on all domain sizes where a comparison is possible, a significant speed-up in comparison to the previous PANDORA version has been achieved.

3.4. Performance of fluid simulations with particles

PANDORA has always been intended to be used for turbulence simulations including Lagrangian inertial particles. In PANDORA 2.0 we have included passive particles so Lagrangian fluid statistics are another application area. With this purpose in mind, the scalability of the code with particles is more essential than the performance of pure fluid simulations. First scalability tests of the new code have been performed on a 256^3 grid with one particle per control volume for strong scaling (left-hand side of figure 6) and on 256^3, 512^3, 1024^3 and 2048^3 domains, also with one particle per control volume, for the weak scaling (right-hand side of figure 6). The strong scaling is close to ideal scal-

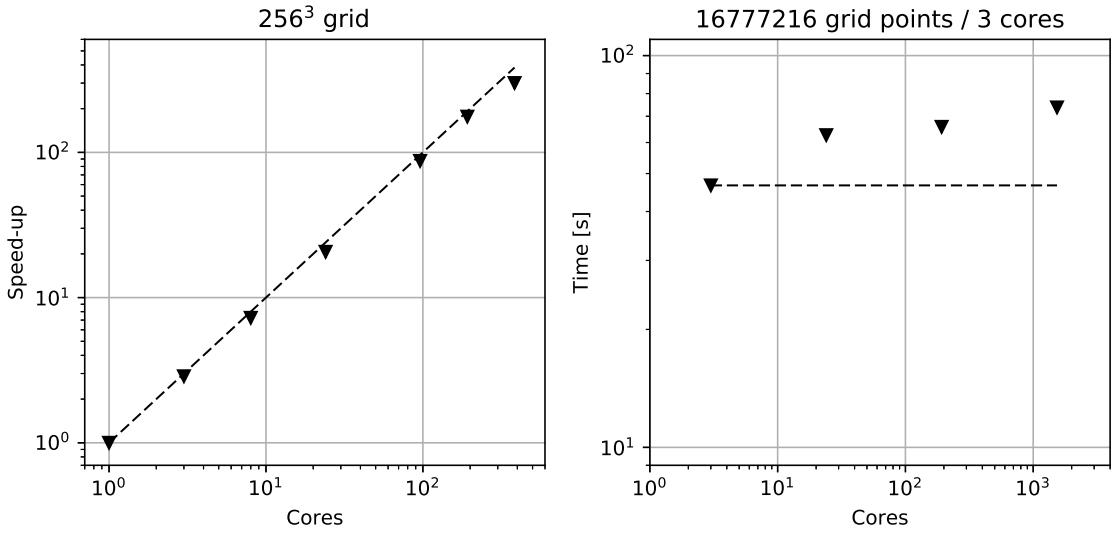

Figure 6: Strong and weak scaling of PANDORA 2.0 for one time step with one particle per control volume. The strong scaling simulations were performed on a 256^3 domain, the weak scaling tests on 256^3, 512^3, 1024^3 and 2048^3 domains. The dashed line indicates ideal scaling.

ability. Only the simulation on 192 cores, corresponding to about 87381 particles per process, shows minimally less than ideal speed-up, indicating that at this point some of the communication starts becoming relevant. The weak scaling tests were performed under exactly the same conditions for all domain-sizes, which means that although the work load per process is about the same, the likelihood of particles moving between different processes increases, therefore leading to more communication. This is reflected in the weak scalability results, which show an increase in simulation times. In practice bigger domains would be typically used to achieve higher Reynolds number. The number of particles moving between processes will also depend on the particles themselves. Therefore the weak scalability will need to be revisited for actual applications.

3.5. Parallel I/O

The parallel file reading and writing routines for fluids and particles are largely identical in PANDORA 2.0, whereas the old code only has a test implementation for particle restart

files. We therefore limited tests of the restart file routines to the fluid information. Table 6 gives an overview of the results. The difference in file size between PANDORA and PANDORA 2.0 is due to the omission of non-zero values. Writing times have been reduced by roughly the time corresponding to the reduction in data volume.

Table 6: Overview of test runs for parallel file writing.

Grid	512^3	1024^3	1024^3	1024^3
Cores	64	24	512	1536
File size [Gb] PANDORA 2.0	0.89	7.13	7.13	7.13
File size [Gb] PANDORA	3.01	-	24.05	-
Time [s] PANDORA 2.0	2.174	63.626	14.809	19.196
Time [s] PANDORA	16.642	-	48.743	-

3.6. First validation runs of the new code

We used the well-known dissipation constant C_ϵ to compare results of PANDORA 2.0 with some published results [7, 9], as shown in figure 7. Our simulations were performed on domain sizes ranging from 128^3 to 1024^3. All results are lower than those obtained by [7], which, as discussed by the authors of that article, could be due to the forcing, which in our simulations was applied to the lowest two wavemodes. These results are consistent with results obtained with the previous PANDORA version [12].

4. Conclusions

A new version of PANDORA has been implemented within this eCSE project. The memory use of the code has been drastically reduced and the parallelisation has been changed from a slab decomposition to a pencil decomposition, which enables the use of the new PANDORA 2.0 on very big simulation domains. So far the code has been successfully tested on simulation grids up to 8192^3. The strong scalability of simulations with particles

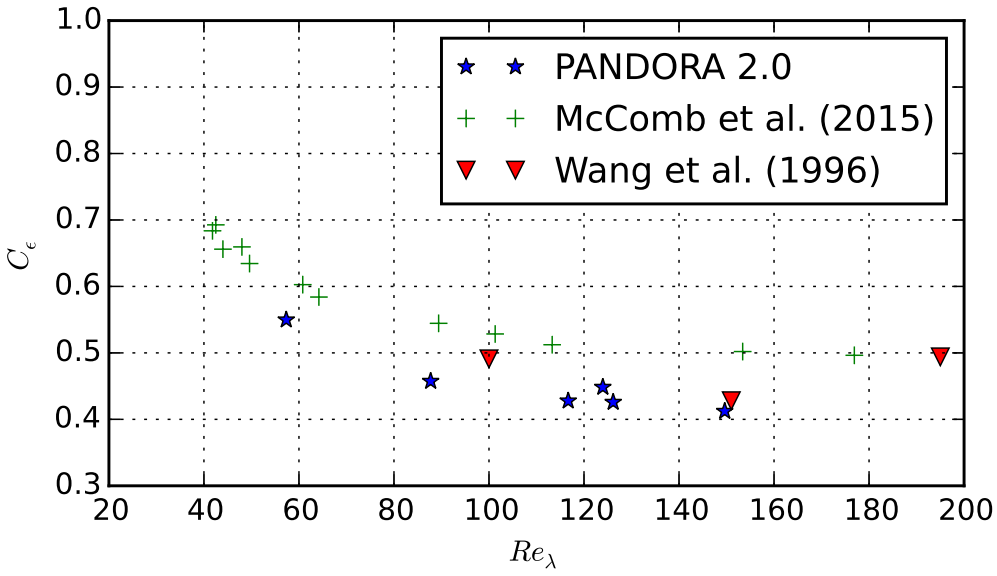

Figure 7: The dissipation constant C_ϵ computed with PANDORA 2.0 compared to published results [7, 9]

has proven to be near ideal scalability. Fluid simulations show reasonably good scaling behaviour within certain limits, which allow for a good performance on all domain sizes.

The use of PETSc for implementing the parallel arrays as well as parallel file reading and writing ensures that future extensions are easier to implement. Possible additions include the implementation of a non-periodic direction, for which the linear algebra routines provided by PETSc might be useful, and an extended choice of file formats, many of which are directly accessible through PETSc. As the code follows the Fortran 2003/2008 standards and extensive use of C data types has been made, both through the PETSc and FFTW libraries and through the ISO C bindings of Fortran, interoperability with C or C++ functions is significantly easier than in the old PANDORA code.

5. Acknowledgements

This work was funded under the embedded CSE programme of the ARCHER UK National Supercomputing Service http://www.archer.ac.uk.

References

[1] Satish Balay, William D. Gropp, Lois Curfman McInnes, and Barry F. Smith. Efficient management of parallelism in object oriented numerical software libraries. In E. Arge, A. M. Bruaset, and H. P. Langtangen, editors, *Modern Software Tools in Scientific Computing*, pages 163–202. Birkhäuser Press, 1997.

[2] Satish Balay, Shrirang Abhyankar, Mark F. Adams, Jed Brown, Peter Brune, Kris Buschelman, Lisandro Dalcin, Alp Dener, Victor Eijkhout, William D. Gropp, Dinesh Kaushik, Matthew G. Knepley, Dave A. May, Lois Curfman McInnes, Richard Tran Mills, Todd Munson, Karl Rupp, Patrick Sanan, Barry F. Smith, Stefano Zampini, Hong Zhang, and Hong Zhang. PETSc Web page. http://www.mcs.anl.gov/petsc, 2018. URL http://www.mcs.anl.gov/petsc.

[3] Satish Balay, Shrirang Abhyankar, Mark F. Adams, Jed Brown, Peter Brune, Kris Buschelman, Lisandro Dalcin, Alp Dener, Victor Eijkhout, William D. Gropp, Dinesh Kaushik, Matthew G. Knepley, Dave A. May, Lois Curfman McInnes, Richard Tran Mills, Todd Munson, Karl Rupp, Patrick Sanan, Barry F. Smith, Stefano Zampini, Hong Zhang, and Hong Zhang. PETSc users manual. Technical Report ANL-95/11 - Revision 3.10, Argonne National Laboratory, 2018. URL http://www.mcs.anl.gov/petsc.

[4] Iain Bethune, Antonia BK Collis, Lennon ON ARAIGH, David Scott, and Prashant Valluri. Developing a scalable and flexible high-resolution dns code for two-phase flows. 2015.

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

[6] Aditya U Karnik. *Direct numerical investigations of dilute dispersed flows in homogeneous turbulence*. PhD thesis, University of Southampton, 2012.

[7] WD McComb, A Berera, SR Yoffe, and MF Linkmann. Energy transfer and dissipation in forced isotropic turbulence. *Physical Review E*, 91(4):043013, 2015.

[8] S. J. Scott. *A PDF Based Method for Modelling Polysized Particle Laden Turbulent Flows Without Size Class Discretization*. PhD thesis, Imperial College London, 2006.

[9] Lian-Ping Wang, Shiyi Chen, James G Brasseur, and John C Wyngaard. Examination of hypotheses in the kolmogorov refined turbulence theory through high-resolution simulations. part 1. velocity field. *Journal of Fluid Mechanics*, 309:113–156, 1996.

[10] JH Williamson. Low-storage runge-kutta schemes. *Journal of Computational Physics*, 35(1):48–56, 1980.

[11] Thorsten Wittemeier. Performance Analysis of the DNS and Particle Transport Model PANDORA. Technical report, University of Southampton, 2016.

[12] Thorsten Wittemeier and John S Shrimpton. Explanation of differences in experimental and computational results for the preferential concentration of inertial particles. *Computers & Fluids*, 173:37–41, 2018.

Appendix - Scalability Data

Table 7: Strong scaling of fluid simulations on ARCHER using PANDORA and PANDORA 2.0. Times are for 10 time steps.

Cores	1	16	32	64	128
Grid	128^3	128^3	128^3	128^3	128^3
Time [s] PANDORA 2.0	9.43	0.85	1.04	0.93	0.85
Speed-up PANDORA 2.0	1.00	11.14	9.07	10.19	11.07
Time [s] PANDORA	25.67	3.36	1.76	1.11	0.56
Speed-up PANDORA	1.00	7.63	14.58	23.10	45.59

Table 8: Strong scaling of fluid simulations on ARCHER using PANDORA 2.0. Times are for 10 time steps.

Cores	1	16	3	64
Grid	128^3	128^3	512^3	512^3
Time [s]	9.43	0.85	261.51	20.32

Table 9: Strong scaling of fluid simulations on ARCHER using PANDORA 2.0 (continued). Times are for 10 time steps.

Cores	24	512	192	1024	1536	2048
Grid	1024^3	1024^3	2048^3	2048^3	4096^3	4096^3
Time [s]	681.13	57.34	926.96	188.95	1278.45	1084.16

Table 10: Weak scaling of fluid simulations on ARCHER using PANDORA and PANDORA 2.0. Times are for 10 time steps.

Cores	1	8	64	512
Grid	128^3	256^3	512^3	1024^3
Time [s] PANDORA 2.0	9.43	12.35	20.32	57.34
Time [s] PANDORA	25.67	49.36	100.44	141.98

Table 11: Weak scaling of fluid simulations on ARCHER using PANDORA 2.0. Times are for 10 time steps.

Cores	3	24	192	1536	12288
Grid	512^3	1024^3	2048^3	4096^3	8192^3
Time [s]	261.52	681.13	926.96	1278.45	2444.87

Table 12: Strong scaling of PANDORA 2.0 for one time step with one particle per control volume.

Cores	1	3	8	24	96	192	384
Grid	256^3	256^3	256^3	256^3	256^3	256^3	256^3
Particles	16777216	16777216	16777216	16777216	16777216	16777216	16777216
Speed-up	1.00	2.87	7.25	20.66	87.29	176.20	300.25
Time [s]	133.91	46.633	18.463	6.481	1.534	0.76	0.45

Table 13: Weak scaling of PANDORA 2.0 for one time step with one particle per control volume.

Cores	3	24	192	1536
Grid	256^3	512^3	1024^3	2048^3
Particles	16777216	134217728	1073741824	8589934592
Time [s]	46.63	62.65	65.78	73.73