An adaptable parallel algorithm for the direct numerical simulation of incompressible turbulent flows using a Fourier spectral/*hp* element method and MPI virtual topologies

A. Bolis[a], C.D. Cantwell[a,*], D. Moxey[a], D. Serson[a], S.J. Sherwin[a]

[a]*Department of Aeronautics, Imperial College London, South Kensington Campus, London, UK*

Abstract

A hybrid parallelisation technique for distributed memory systems is investigated for a coupled Fourier-spectral/*hp* element discretisation of domains characterised by geometric homogeneity in one or more directions. The performance of the approach is mathematically modelled in terms of operation count and communication costs for identifying the most efficient parameter choices. The model is calibrated to target a specific hardware platform after which it is shown to accurately predict the performance in the hybrid regime. The method is applied to modelling turbulent flow using the incompressible Navier-Stokes equations in an axisymmetric pipe and square channel. The hybrid method extends the practical limitations of the discretisation, allowing greater parallelism and reduced wall times. Performance is shown to continue to scale when both parallelisation strategies are used.

Keywords: Spectral/hp element method, High-order methods, incompressible flows, MPI parallelisation, virtual topologies

1. Introduction

Direct Numerical Simulation (DNS) is used to simulate complex laminar and turbulent flow problems both to gain an understanding of the fundamental flow physics and for industrial applications [1]. Turbulent flows inherently require high spatial and temporal resolutions in order to resolve the spectrum of scales within the flow, which depends on the Reynolds number *Re*. The number of grid points needed to resolve a fully turbulent three-dimensional flow scales as $Re^{9/4}$ [2], meaning that even at modest *Re*, the computational demands are significant. Since practical CFD applications involve Reynolds numbers on the order of $10^3 - 10^6$ and higher, this inevitably makes serial computation impossible, necessitating the use of parallel clusters of computers.

The most prevalent form of high-performance computer systems are distributed-memory clusters consisting of an interconnected collection of processors, each with their own local memory hierarchies. Traditionally, the capacity of these systems has been broadly increased through faster processor clock speeds and improved lower-latency network interconnects. However, in recent years, HPC facilities have evolved around increasingly parallel systems as clock speeds have saturated and energy-usage concerns become a motivating factor [3]. Consequently, algorithms have been required to adapt to the changing hardware landscape in order to maintain efficiency.

The spectral/*hp* element method [4], whilst being used to simulate fluid flow for many years in an academic setting, is now emerging as an attractive alternative to many traditional numerical discretisations on modern HPC hardware. As opposed to the classical finite element method, spectral/*hp* elements use high-order polynomial expansions on each element. Numerically, this has the advantage of low dispersion and diffusion alongside exponential convergence in the polynomial order. Additionally however, discretised operators are dense and have a far richer structure compared to linear expansions, meaning that they can more effectively utilise caching on modern HPC hardware. The tensor product of one-dimensional basis functions on each element also admits a rich fabric of implementation strategies [5, 6, 7].

However, variations of this method mean that we can further improve computational performance whilst preserving the accuracy of the simulation. Many studies of fundamental flow physics are posed on domains which are characterised by geometric homogeneity in one or more coordinate directions [8, 9, 10]. Instead of discretising the domain using a 3D spectral/*hp* element method, one can combine a 2D spectral/*hp* element discretisation with a pure spectral expansion to significantly reduce the computational cost of these problems. This approach is known as a *Fourier–spectral*/hp *element method* [11]. In this study we are specifically interested in the case where only one coordinate direction posesses geometric homogeneity. Therefore, the 3D domain is decomposed into a sequence of spectral/*hp* element planes, coupled using a Fourier expansion in the third coordinate directions.

The approach typically used when parallelising this type of discretisation is to either use mesh-decomposition in the spectral/*hp* element planes, or apply a modal decomposition in the Fourier direction. The latter takes advantage of the orthogonality property of the Fourier basis for linear operators. The optimal choice of parallelisation strategy typically depends on the size of the problem, the ratio of Fourier planes to spectral elements, alongside the hardware and interconnect of the parallel system. Moreover, the fast development of computer systems forces software designers to make a continuous effort to main-

*Corresponding author (e-mail: c.cantwell@imperial.ac.uk)

tain algorithms to be able to exploit all the benefits exposed by the latest generation of hardware. There is therefore benefit to be gained from a code supporting both types of parallelism, but predicting the performance of these algorithms on a specific architecture is not trivial [12].

The performance of a parallel 3D incompressible Navier-Stokes solver using the Fourier–spectral/*hp* element method has been benchmarked previously [13, 14]. Spectral modes were distributed across the processes, requiring the transposition of data using the MPI all-to-all technique to compute derivatives in that direction. Their performance model assumed a flat communication topology and the maximum number of processes was limited to the number of Fourier planes. Conversely, parallelisation of a spectral element discretisation has been explored [15, 16, 17, 18, 19], for which the upper limit on the number of processes is the number of mesh elements.

Performance of a mixed-parallelism case for 3D turbulence simulations has previously been investigated [20], specifically for a 1D spectral/*hp* element discretisation, coupled with a 2D spectral expansion. Parallel communication was implemented across processes as a Cartesian topology and a performance model was constructed which suggested improved strong scaling could be achieved on specific architectures. Solver performance depends on hardware characteristics such as memory bandwidth and processor cache size, but also on network capabilities in terms of latency, and bandwidth. Therefore prudent choice of parallelism strategy can enable improved overall performance by structuring the computation and communication pattern to better match the available hardware.

The present study is distinguished from this previous work by the choice of discretisation. We use a two-dimensional spectral/*hp* element method, coupled with a one-dimensional spectral expansion. This permits the investigation of flow problems on geometries of significantly greater complexity than earlier works. We first outline the discretisation and parallelisation strategies and quantify their comparative performance. For large runs with many processes the number of possible hybrid parallelism strategies may be significant. We construct a mathematical model to characterise the expected performance of any given single or hybrid parallelisation strategy which can be used to predict the optimal strategy for a given problem. We calibrate the model against the individual mesh-decomposition and Fourier parallelisation techniques and demonstrate its accuracy in predicting performance of the hybrid approach.

2. Methods

Three-dimensional incompressible, isothermal flow with constant density and viscosity is governed by the incompressible Navier-Stokes equations which, in terms of the primitive variables $(\boldsymbol{u}, p)$, are expressed as

$$\frac{\partial \boldsymbol{u}}{\partial t} + \boldsymbol{u} \cdot \nabla \boldsymbol{u} = -\nabla p + \nu \nabla^2 \boldsymbol{u},$$
$$\nabla \cdot \boldsymbol{u} = 0,$$

where p is the kinematic pressure, ν is the kinematic viscosity and $\boldsymbol{u} = [u, v, w]^\top$ is the velocity.

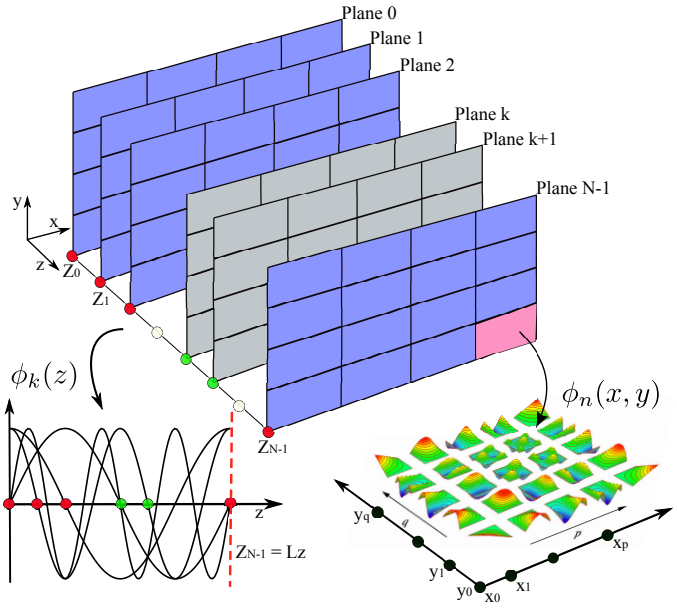

Figure 1: Structure of a three-dimensional expansion using a Fourier spectral/*hp* element method.

2.1. Spatial discretisation

The three-dimensional domain is decomposed into N_Z two-dimensional spectral/*hp* element planes spanning the x and y coordinate directions, coupled with a Fourier expansion in the third homogeneous direction, as illustrated in Figure 1. The spectral/*hp* element discretisation is described elsewhere [4] and only a brief summary is given here.

Each two-dimensional plane Ω_k is partitioned into a set of N_{el} subdomains Ω_k^e such that,

$$\Omega_k = \bigcup_{e=0}^{N_{\text{el}}} \Omega_k^e$$
$$\Omega_k^e \cap \Omega_k^f = \emptyset \quad \forall e \neq f.$$

In this study the meshes consist of quadrilateral elements only, but the approach may be equally applied when using triangular elements. Numerical integration and differential operators are constructed on a standard reference element Ω^{st} which is mapped to each Ω_k^e using a bijective map, $\chi^e : \Omega^{\text{st}} \to \Omega^e$, as $\boldsymbol{x} = \boldsymbol{\chi}^e(\boldsymbol{\xi})$. On each element, the solution u may be approximated as

$$u^\delta(x, y) = \sum_{n=0}^{N} \phi_n(x, y)\hat{u}_n = \sum_{p=0}^{P}\sum_{q=0}^{P} \phi_p(x)\phi_q(y)\hat{u}_{pq}^e,$$

where $\hat{u}_{pq}^e$ are elemental coefficients. These correspond to the tensor-product of nodal expansion bases, $\phi_p(x)$ and $\phi_q(y)$, of order P defined as Lagrange polynomials through Gauss-Lobatto-Legendre points ξ_i, and have the form

$$\phi_m(\xi) = \frac{\prod_{l=0, l\neq m}^{P}(\xi - \xi_l)}{\prod_{l=0, l\neq m}^{P}(\xi_m - \xi_l)}.$$

This is synonymous with the original spectral element method, giving a total of $(P+1)^2$ degrees of freedom (DOF) per element and $N_{XY} = N_{el} \times (P+1)^2$ local degrees of freedom per plane. Gaussian quadrature is used for numerical integration, for which the solution $\mathbf{u}$ is represented on the same set of $P+1$ points ξ_m.

The connectivity of elements in a plane is represented by an assembly mapping $\mathcal{A}$ which maps the concatenated vector of elemental degrees of freedom to their global counterparts and enforces a C^0-continuity constraint. The global degrees of freedom are assembled using the relation $\hat{\mathbf{u}}^{\mathbf{e}} = \mathbf{A}\hat{\mathbf{u}}^{\mathbf{g}}$, where $\mathbf{A}$ is the matrix equivalent of $\mathcal{A}$. This matrix is in general highly sparse and so is in practice not constructed explicitly.

Operators in the spectral/hp element method are constructed elementally and applied using the sum-factorisation technique [21] as this has been demonstrated to be more efficient when operating on elements with higher-order bases [7, 6, 5]. The tensor-product nature of the elemental expansion bases allows matrix-vector operations to be decomposed into a sequence of smaller, more computationally efficient matrix-matrix operations, performed in each coordinate direction separately.

In the z-direction, the solution is expressed using a Fourier basis of $N_Z/2$ complex modes, $\phi_k(z) = e^{ izk}$, to give an expansion of the three-dimensional solution on an element as

$$u^{\delta}(x,y,z) = \sum_n \phi_n(x,y,z)\hat{u}_n = \sum_{pqk} \phi_{pq}(x,y)\phi_k(z)\hat{u}_{pqk}.$$

The total number of degrees of freedom is therefore $N_{\text{tot}} = N_{XY}N_Z$.

2.2. Temporal discretisation

A stiffly stable splitting scheme [22] is adopted which decouples the velocity and pressure fields, leading to an explicit treatment of the advection term and an implicit treatment of the pressure and the diffusion terms. The key steps are

$$\begin{aligned}
\frac{\bar{\boldsymbol{u}} - \sum_{q=0}^{J-1} \alpha_q \boldsymbol{u}^{n-q}}{\Delta t} &= -\sum_{q=0}^{J-1} \beta_q [(\boldsymbol{u}\cdot\nabla)\boldsymbol{u}]^{n-q}, \\
\nabla^2 p^{n+1} &= \nabla \cdot \left(\frac{\bar{\boldsymbol{u}}}{\Delta t}\right), \\
\frac{\bar{\bar{\boldsymbol{u}}} - \bar{\boldsymbol{u}}}{\Delta t} &= -\nabla p^{n+1}, \\
\frac{\gamma_0 \boldsymbol{u}^{n+1} - \bar{\bar{\boldsymbol{u}}}}{\Delta t} &= \nu \nabla^2 \boldsymbol{u}^{n+1}.
\end{aligned}$$

To maintain the order of the scheme, a modified Neumann pressure boundary condition is used,

$$\frac{\partial p}{\partial n}^{n+1} = -\left[\frac{\partial \boldsymbol{u}}{\partial t}^{n+1} + \nu \sum_{q=0}^{J-1} \beta_q (\nabla \times \omega)^{n-q} + \sum_{q=0}^{J-1} \beta_q [(\boldsymbol{u}\cdot\nabla)\boldsymbol{u}]^{n-q}\right] \cdot \boldsymbol{n}.$$

The coefficients α_q, β_q and γ_0 can be found in [22] for first-, second- and third-order implicit-explicit (IMEX) time-integration schemes. Figure 2 illustrates the implementation of the time-integration section of the algorithm, where we ignore input/output and set-up costs. For short time-integration, these may be significant.

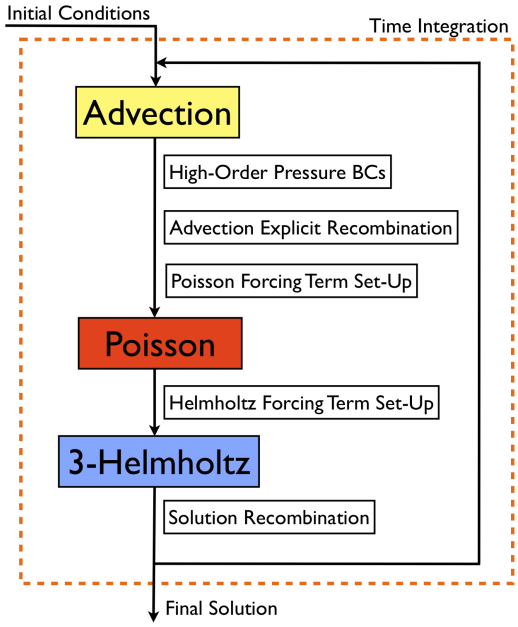

Figure 2: Incompressible Navier-Stokes solution algorithm. Details of the building blocks of the time-integration process. The most expensive routines are highlighted, *i.e.* the advection term calculation and the elliptic solvers for pressure and velocity (Poisson and Helmholtz).

2.3. Parallelisation

In this section we describe the parallelisation approaches used in this study. We provide an overview of the two orthogonal approaches: parallel decomposition of the Fourier modes (modal parallelisation) and parallel decomposition of the spectral/hp element planes (elemental parallelisation). We then outline the hybrid approach which combines both techniques, and describe its implementation. In each case we partition the simulation across a total of R processes. The Message Passing Interface (MPI) library is used for communication in all three methods.

In modal parallelisation the N_Z planes, corresponding to $N_Z/2$ complex Fourier modes, are distributed equally among the processes. Elliptic solves are decoupled in the Fourier-transformed space and can be performed independently on each plane using either a direct Cholesky factorisation with reverse Cuthill-McKee algorithm (LAPACK) [23], or through an iterative conjugate gradient algorithm. The non-linear advection term is more efficiently computed in non-modal space. To perform the inverse and forward Fourier transforms, used before and after the advection calculation respectively, the data to be transformed must reside on the same process. In practice, this requires a transposition of the data using an MPI all-to-all operation. To support efficient differentiation in the z-coordinate direction, we additionally impose the constraint that both the real and imaginary components of each complex Fourier mode

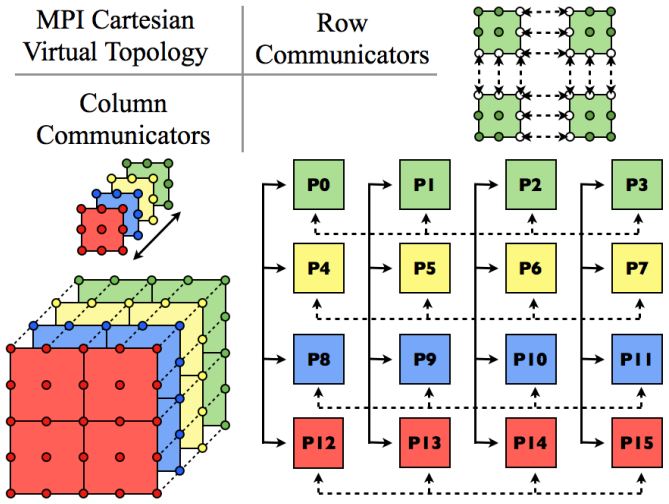

Figure 3: Illustrative MPI cartesian communicator for a hybrid parallelisation of a Fourier spectral/*hp* element discretisation using 4 elements per plane and 4 planes, on 16 MPI processes. Row communicators handle the communication between mesh partitions for elemental parallelisation while column communicators handle communication between planes for modal parallelisation.

reside on the same process, since, in the Fourier space, derivatives are calculated through the multiplication $\hat{u}_k \mapsto -ik\hat{u}_k$. This restricts the maximum number of usable processes to $N_Z/2$.

In contrast, elemental parallelism distributes the N_{el} elements of each plane among the processes. The partitioning of the 2D plane is implemented using the METIS graph partitioning library [24] and an identical partitioning and distribution amongst processes is used for each plane in the domain. The natural limit on the number of usable processes is therefore N_{el}. The dual-graph of the mesh is partitioned among the R processes to equally distribute the number of degrees of freedom, whilst minimising the edge-cut, and therefore the inter-process communication. Elliptic solves are performed iteratively, with communication being required to exchange boundary information between adjacent elements residing on different processes at each iteration. This data exchange is implemented using the gather-scatter algorithm from Nek5000 [25] which uses a global numbering of the DOFs in the plane to efficiently summate process-local contributions and distribute the result back to the participating processes.

Hybrid parallelisation combines both modal and elemental approaches by organising the available processes in a Cartesian grid [20], as illustrated in Figure 3. In this arrangement, the world communicator is split into a series of row communicators which support elemental parallelisation, while column communicators enable modal parallelisation. Each process belongs to precisely one row communicator and one column communicator and nominally operates on a fixed subset of elements in a fixed subset of planes. As in modal parallelism, elliptic solves are performed in Fourier-transformed space, but due to the elemental parallelism the iterative conjugate gradient solver must be used. The limit on the number of viable processes is now increased substantially to $N_{el} \times N_Z/2$.

2.4. Test Environment

All simulations are performed on an SGI Altix ICE 8200 EX system with up to 512 cores (64 eight-core nodes). Each node contains Nehalem CPUs running at 2.93 GHz and 24GB RAM. Communication is through a dual-rail *Infiniband* interconnect. The system runs Redhat Enterprise Linux with kernel version 3.0.58-0.6.6. Intel MPI was used for parallel message exchange and FFTW 3.2.2 for performing fast Fourier transforms.

The software used for the spectral/*hp* element discretisation in this study was Nektar++ v3.3.0 [26, 27]. In summary, the framework provides scope for constructing high-order polynomial expansions on both fully two-dimensional and three-dimensional domains. It also supports a coupled spectral/*hp* element – Fourier approach for domains with geometric homogeneity. The specific operators and time integration necessary for solving the incompressible Navier-Stokes equations are built upon this framework. As with any numerical timing study, the presented results are specific to the implementation used, although should provide useful generic guidance.

3. Performance Model

Identifying the optimal strategy and the distribution of processes between elemental and modal parallelism is non-trivial, since algorithmic complexity and specific system architecture affects performance. We therefore design a performance model, calibrated through the use of the two parallelisation strategies independently, to help select the best approach before the start of a given simulation.

3.1. General model assumptions

To ensure the model remains simple enough for predictive use, yet sufficiently complex to provide reasonable accuracy, we make a number of assumptions regarding the nature of the computational problem and hardware when evaluating the computational and communication costs.

The computational cost of an algorithmic unit is evaluated using the floating-point operation count of basic routines such as matrix-vector multiplications, inner products and vector-vector summations. This implicitly disregards hardware characteristics, such as caching and memory throughput limits, although our testing has shown that these aspects can be reasonably captured using scalar constants, determined during the calibration process for a specific platform. Operations are evaluated at the element level, and their total computational cost across the domain is therefore assumed to be predominantly independent of the parallelisation strategy.

Communication costs are generally more complex to model and strongly depend on the hardware configuration. Different cluster configurations, such as mesh, hypercube or ring interconnect topologies, have a significant effect on the measured communication time. In this study we follow the most common approach when estimating communication costs [20, 14], which is to assume a "flat" topology supporting direct communication between nodes and no interconnect contention. Operations are assumed to be performed using double-precision floating point

numbers, occupying eight bytes on the test system described above.

3.2. Model construction

The general structure of our performance model is as follows. Let O_i be the operation count of the i-th operation in the algorithm. Let C_j be the time required for the $j-th$ communication, then we can define the total parallel execution time, T, as

$$T = \frac{1}{R_{XY}R_Z}\sum_i O_i + \sum_j C_j.$$

where R_{XY} and R_Z are the number of processes used for elemental and Fourier parallelism, respectively, and $R = R_{XY}R_Z$. The size of a computational problem is generally measured by the number of degrees of freedom. In the spectral/*hp* discretisation, this corresponds to the number of elemental modes which, for two-dimensional quadrilateral elements, is $(P+1)^2$. For some matrix-vector operations the sum-factorisation technique, which exploits the tensorial nature of the expansion, can be used that requires $(4P^3 + 18P^2 + 26P + 12)$ operations per element [7].

The communication times T_{Cj} can be further modelled as

$$C_j = N_{\text{msgs}} \times \left[\tau_L + N_{\text{op}} \times \tau_B\right] \tag{2}$$

where N_{msgs} is the number of messages transmitted during an operation, N_{op} is the number of floating-point values per message, τ_L is the latency and τ_B the inverse of the bandwidth. Note that τ_B is quantified using s/DOF, rather than the conventional s/MB, to facilitate the modelling. Latency and bandwidth are sampled for the test system using the MPI benchmarking application IBM-MP1. Bandwidth is measured for a number of MPI routines and for messages of size 8 bytes up to 4MB and averaged. For the test system considered, the average bandwidth measured was $1.64 \cdot 10^3\ MB/s$. Bandwidth and latency for the test system was determined to be

$$\tau_B = 4.87 \cdot 10^{-9}\ s/DOF, \qquad \tau_L = 2.09 \cdot 10^{-6}\ s.$$

3.3. Advection Term

We first model the advection term $\boldsymbol{u} \cdot \nabla \boldsymbol{u}$, which is computed in physical space and can be expanded as

$$\begin{aligned} N(u) &= u\partial u/\partial x + v\partial u/\partial y + w\partial u/\partial z, \\ N(v) &= u\partial v/\partial x + v\partial v/\partial y + w\partial v/\partial z, \\ N(w) &= u\partial w/\partial x + v\partial w/\partial y + w\partial w/\partial z, \end{aligned}$$

where u, v and w denote the three components of the velocity $\boldsymbol{u}$. The numerical implementation of this is shown in Algorithm 1. Computational costs arise from FFTs (lines 1, 4 and 6), derivatives (lines 2 and 3) and vector-vector operations (line 5), while communication is only required to compute the FFTs.

Inverse FFTs are required for each of the velocity components and the z-derivatives of each of the velocity components. A forward FFT is used to transform the result of the advection calculation. This gives a total of nine FFTs, each consisting

input : $\tilde{u}_0, \tilde{u}_1, \tilde{u}_2$

output: $\tilde{N}(u_0), \tilde{N}(u_1), \tilde{N}(u_2)$

`// Transformation back to physical space`

for $i = 0$ **to** 2 **do**

1 $u_i = IDFT(\tilde{u}_i)$;

end

for $i = 0$ **to** 2 **do**

`// Derivatives in the 2D spectral/hp element plane`

2 $\partial u_i/\partial x = \boldsymbol{D_x} u_i \qquad \partial u_i/\partial y = \boldsymbol{D_y} u_i$;

`// Derivatives in the spectral direction`

3 $\partial \tilde{u}_i/\partial z = \tilde{\boldsymbol{D}}_z \tilde{u}_i$;

`// Transformation back to physical space`

4 $\partial u_i/\partial z = IDFT(\partial \tilde{u}_i/\partial z)$;

`// Construction of the` $i-th$ `advection component`

5 $N(u_i) = u_0\partial u_i/\partial x + u_1\partial u_i/\partial y + u_2\partial u_i/\partial z$;

`// Transformation to Fourier space`

6 $\tilde{N}(u_i) = DFT(N(u_i))$;

end

Algorithm 1: Non-linear advection term procedure. Terms with a tilde are forward Fourier-transformed.

of a set of independent 1D FFTs. The number of 1D FFTs is given by the number of quadrature points associated with the local spectral/*hp* element mesh partition. Assuming the mesh is evenly partitioned, we can quantify the total number of 1D FFTs as $N_{el}(P+1)^2$, each costing $O(N_Z \log_2(N_Z))$, giving a cumulative cost of

$$O_1^A = 9N_{el}(P+1)^2 \cdot C_{\text{FFT}} N_Z \log_2 N_Z,$$

where C_{FFT} is a const. Derivatives in the z-direction are a BLAS level 1 operation with total cost

$$O_2^A = N_{el}N_Z(P+1)^2$$

In-plane physical derivatives will be proportional to the cost of executing a matrix-vector multiplication using the general derivative matrix. A total of six in-plane derivatives are computed. The total cost of derivative operations is therefore,

$$O_3^A = 6N_{el}(P+1)^4.$$

Finally, there are five level 2 BLAS operations in calculating each component of $N(u_i)$. Vectors are of size $N_{el}(P+1)^2$, resulting in a total cost of

$$O_4^A = 15N_{el}(P+1)^2.$$

For the advection term, communication is required during the 9 FFTs to shuffle data between processes so that the data for each 1D FFT, previously spanning R_Z processes, is colocated on the same process. We apply the communication model described in (2). For each of the 9 FFTs two MPI All-to-all calls

input : initial guess $\boldsymbol{x}_0$

output: final solution $\boldsymbol{x}$

`// calculate initial residual` $\boldsymbol{r}_0$

$\boldsymbol{r}_0 = \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}_0$;

`// solve for` $\boldsymbol{w}_0$ `where` $\boldsymbol{K}$ `is the preconditioner`

$\boldsymbol{K}\boldsymbol{w}_0 = \boldsymbol{r}_0$;

`// set parameters`

$\boldsymbol{q}_{-1} = \boldsymbol{p}_{-1} = 0 \qquad \beta_{-1} = 0 \qquad \boldsymbol{s}_0 = \boldsymbol{A}\boldsymbol{w}_0$;

$\rho_0 = (\boldsymbol{r}_0, \boldsymbol{w}_0) \qquad \mu_0 = (\boldsymbol{s}_0, \boldsymbol{w}_0) \qquad \alpha_0 = \rho_0/\mu_0$;

for $i = 0$ **to** $\mathcal{N}_{iter}^{MAX}$ **do**

1 $\quad \boldsymbol{p}_i = \boldsymbol{w}_i + \beta_{i-1}\boldsymbol{p}_{i-1}$;

2 $\quad \boldsymbol{q}_i = \boldsymbol{s}_i + \beta_{i-1}\boldsymbol{q}_{i-1}$;

3 $\quad \boldsymbol{x}_{i+1} = \boldsymbol{x}_i + \alpha_i\boldsymbol{p}_i$;

4 $\quad \boldsymbol{r}_{i+1} = \boldsymbol{r}_i - \alpha_i\boldsymbol{q}_i$;

$\quad$ **if** $(\boldsymbol{r}_{i+1}, \boldsymbol{r}_{i+1}) < tolerance$ **then**

$\quad\quad$ break;

$\quad$ **end**

5 $\quad$ Solve $\quad \boldsymbol{K}\boldsymbol{w}_{i+1} = \boldsymbol{r}_{i+1}$;

6 $\quad \boldsymbol{s}_{i+1} = \boldsymbol{A}\boldsymbol{w}_{i+1}$;

7 $\quad \rho_{i+1} = (\boldsymbol{r}_{i+1}, \boldsymbol{w}_{i+1})$;

8 $\quad \mu_{i+1} = (\boldsymbol{s}_{i+1}, \boldsymbol{w}_{i+1})$;

$\quad \beta_i = \rho_{i+1}/\rho_i$;

$\quad \alpha_{i+1} = \rho_{i+1}/(\mu_{i+1} - \rho_{i+1}\beta_i/\alpha_i)$;

end

Algorithm 2: Preconditioned Conjugate Gradient Method. *Demmel et al.* [28].

are required (shuffling and unshuffling), each of which formally requires $N_{\text{msgs}} = (R_Z - 1)$ messages [20, 14]. Message size is based on the assumption that the 1D FFTs will be evenly distributed across the participating processes. This gives a communication cost of

$$C_1^A = 18(R_Z - 1)\left(\tau_L + \frac{N_{el}}{R_Z R_{XY}}(P+1)^2\tau_B\right).$$

Combining the above contributions and distributing the computational cost amongst the processes gives a parallel execution time of

$$T^A = \frac{1}{R_Z R_{XY}} \cdot \sum_i O_i^A + \sum_j C_j^A.$$

3.4. Elliptic Solver

Algorithm 2 shows the basic steps to solve the linear systems using a preconditioned conjugate gradient method. To simplify the analysis, we do not perform static condensation of the elliptic systems, evaluating them using a block-diagonal matrix system, where each block contains a full elemental matrix.

The `daxpy` operations on lines 1-4, each comprising one scalar-vector multiplication and one vector-vector summation giving a total cost of

$$O_1^E = 8N_{el}(P+1)^2.$$

The application of the diagonal preconditioner in step 5 can be considered a vector-vector multiplication and has cost

$$O_2^E = N_{el}(P+1)^2.$$

The most computationally expensive step is the evaluation of the matrix system in step 6. Applying the sum-factorisation operation count defined earlier in this section we quantify the number of operations as

$$O_3^E = N_{el}(4P^3 + 18P^2 + 26P + 12).$$

Finally, the two inner products in steps 7 and 8 evaluate the stopping criteria of the iterative algorithm. Each consist of a vector-vector multiplication and a sum reduction. The vector-vector multiplication requires $N_{el}(P+1)^2$ operations per plane while the sum reduction $N_{el}(P+1)^2 - 1$ operations per plane. In order to maintain simplicity in the model we approximate the sum reduction to $N_{el}(P+1)^2$ operations, leading to

$$O_4^E = O_8^E = 6N_{el}(P+1)^2.$$

Communication appears during the inner product reductions and during the matrix-vector multiplication. The inner product reduction can be modelled using the All-gather model [20]. The number of messages is $(R_{XY} - 1)$ for each inner product, since the local reductions need to be composed into a global reduction, which happens on one processor. Since the reduction operations for each iteration can be collated into a single message, the size is three floating-point values. This results in the communication time

$$C_1^E = (R_{XY} - 1)\Big(\tau_L + 3\tau_B\Big).$$

To estimate communication during matrix-vector multiplication, we assume the structure of the mesh decomposition leads to a tree-like graph of communication, arising from recursive bisection. The number of communications will therefore be proportional to $\log_2(R_{XY})$. Furthermore, we can assume data needs to be exchanged in both directions for each edge of the tree. Message size is far more challenging to estimate, since this is dependent on the size of the boundary between any two partitions. We therefore choose the maximum message size, which can be estimated at $2(N_{el}^{loc} + 1)$, as illustrated in Figure 4. Here we are also assuming that all partitions are interior to the domain and therefore all boundaries participate in communication. These estimates lead to a prediction for the matrix-vector multiplication communication costs as

$$C_2^E = 2C_{GS}\log_2(R_{XY})\left[\tau_L + 2\left(\frac{N_{el}}{R_Z R_{XY}} + 1\right)(P+1)\tau_B\right],$$

where C_{GS} is a constant relating to the implementation of the gather-scatter algorithm.

Combining these contributions gives the cumulative cost of a single iteration of the elliptic solver as

$$T^E = \frac{1}{R_Z R_{XY}} \cdot \sum_i O_i^E + \sum_j C_j^E.$$

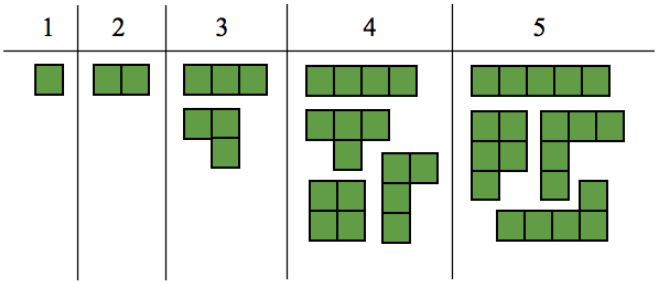

Figure 4: Overview of how a partition containing N_{el}^{loc} can be cast. The different groupings suggest that the maximum number of edges which may require communication is $\propto 2(N_{el}^{loc}+1)$

3.5. Incompressible Navier-Stokes Model

We now combine the above components of the model to ellicit a full model for the incompressible Navier-Stokes algorithm described in Figure 2. The total parallel execution time for one time-step can be expressed as

$$T^{NS} = a \cdot T^{A} + b \cdot \left(\mathcal{N}_{iter}^{P} + 3\mathcal{N}_{iter}^{H}\right) \cdot T^{E}, \tag{3}$$

which captures the costs associated with the advection term, Poisson solve for the pressure and the three Helmholtz solves for the velocity components. $\mathcal{N}_{iter}^{P}$ and $\mathcal{N}_{iter}^{H}$ are the number of iterations of the Poisson and Helmholtz solves, respectively. These will vary depending on the nature of the problem and the choice of preconditioner plays an important role in the efficiency of the iterative solver. The diagonal preconditioner was chosen for modelling simplicity and is not necessarily the most efficient choice. Typical values are $\mathcal{N}_{iter}^{P} \sim 80$ and $\mathcal{N}_{iter}^{H} \sim 10$ for the problems considered in this study. However, these are problem-specific and are largely independent of the parallelisation strategy. The coefficients a and b capture the characteristics of specific hardware and are determined during the calibration process discussed next.

4. Results

We consider two prototype turbulent flow problems to quantify the performance of the different parallelisation regimes. These examples highlight the benefits of the hybrid parallelisation approach for increasing parallelism in a scalable way and therefore reducing parallel execution time. Table 1 lists the performance model properties of the two domains considered. Since this study concerns only the parallelisation aspect of these simulations, we consider a fixed discretisation in each case. The discretisation is chosen to be numerically converged for capturing turbulent flow in the given geometry and at the prescribed Reynolds number, based on previously published studies [10, 29, 30].

4.1. Test problems

The pipe geometry is illustrated in Figure 5, where the streamwise direction is geometrically homogeneous. Lengths and velocities are non-dimensionalised by the diameter D and the bulk velocity u_{bulk}, respectively. The length of the pipe is $5D$. The flow is driven by a constant body-force of

$$f_z = 0.5 * 0.3164/\text{Re}^{0.25}$$

for $Re = 3000$ to instil a turbulent flow regime. The pipe is discretised using spectral elements in the cross-section of the pipe and a Fourier expansion in the streamwise direction. A total of 64 spectral elements at polynomial order 7 are used in the x-y plane, while 128 modes are used in the Fourier expansion. No-slip boundary conditions are imposed on the wall of the pipe. The time-step used for simulations is $\Delta t = 0.002$ non-dimensional time units with a second-order IMEX scheme.

Table 1: Turbulent test cases discretisation features.

Test case	P	N_{el}^{plane}	N_Z	N_{el}	N_{XY}	N_{TOT}
Pipe	7	64	128	8192	5184	663552
Channel	6	450	64	28800	28800	1843200

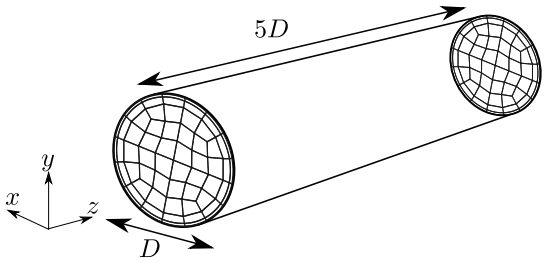

Figure 5: Diagram illustrating the pipe geometry and its discretisation. Spectral/hp elements are used in the cross-plane with a Fourier expansion used in the streamwise direction.

For the channel, shown in Figure 6, lengths are non-dimensionalised by the channel half-height and velocities by u_{bulk}. The length of the channel is 4π, and the spanwise dimension is $4\pi/3$. The flow is driven by a body force of $f_x = 0.0036$ for $Re = 3000$ and no-slip boundary conditions are imposed on the top and bottom of the channel. The channel was discretised using 64 Fourier modes and 450 spectral elements with a polynomial order of 6. The time-step used for channel flow simulations was 0.0001 with a second-order IMEX scheme.

4.2. Hybrid parallelism performance

The efficiency of the various parallel strategies is assessed through the strong scaling tests for both problems. The results for the pipe are shown in Figure 7. Modal parallelism (triangle and square symbols) scales well using either direct or iterative elliptic solvers. Elemental parallelism (circle symbols) scales poorly due to the large ratio of communication to computation, since even for only 32 processes, there are only 2 elements per process. The dotted lines indicate theoretical bottlenecks on the number of usable processes due to there being an insufficient number of elements or Fourier modes. In both cases, this limit is 64 processes. However, in the case of a Fourier-dominanted discretisation, modal parallelism is clearly preferably over elemental parallelism.

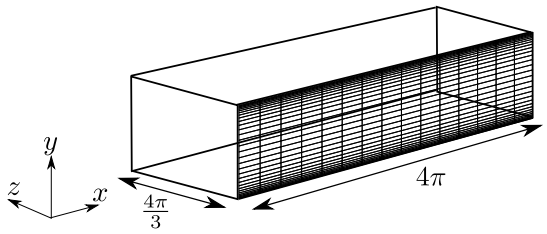

Figure 6: Diagram illustrating the channel geometry and its discretisation. A Fourier expansion is used in the spanwise direction.

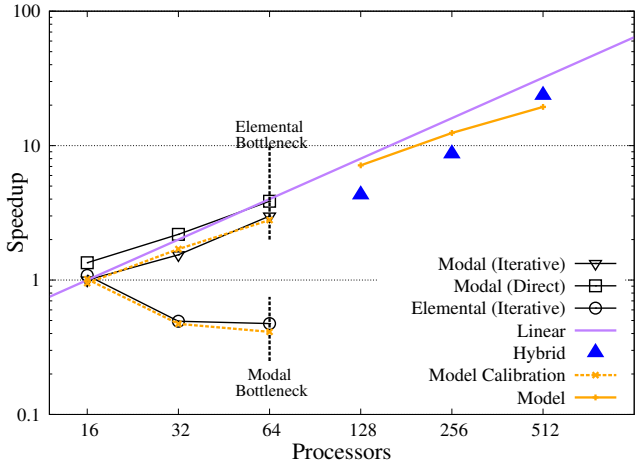

Figure 7: Parallel efficiency of the four parallel approaches for the pipe flow problem on the system detailed in Section 2.4. The vertical dotted black lines indicates the practical limit imposed by the modal and elemental discretsations in isolation. The solid purple line shows ideal efficiency, relative to the 16-core case using modal parallelism with iterative elliptic solver. The dotted orange lines show the model prediction for the cases used for calibration. The solid orange lines show predictions for the hybrid regime.

Figure 8 shows a comparison of efficiency for the different parallelism strategies in the channel problem. Here the modal bottleneck is reached at 32 cores and the modal approach with direct solver has the greatest performance in this regime. Elemental parallelism is possible up to 450 cores but, above 128 cores, the ratio of computation to communication is low and at 256 cores, the distribution of computation becomes significantly unequal, resulting in poor parallel performance. However, elemental parallelism outperforms modal parallelism when using the iterative solver. This observation is intuitive since only four nodes are used and most of the communication between partitions will be intra-node. Recent versions of the OpenMPI libraries allow processes on the same node to use shared memory, rather than using the network interface to send messages. Within a node latency between processes is therefore low and sending a large number of small messages becomes the most effective method.

Hybrid parallelism extends these limits substantially, enabling simulations up to and beyond 512 processes. The distribution of modal and elemental parallelism will lead to different performance. In Figure 7 and Figure 8 the solid triangles indicate the minimum execution time achievable using hybrid parallelism for a prescribed total number of cores. Efficiency

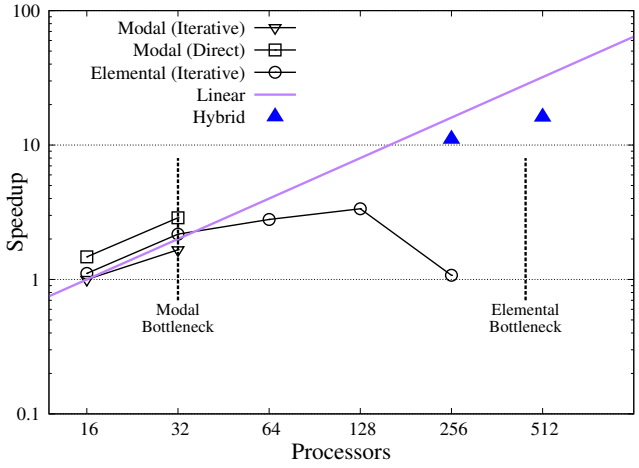

Figure 8: Parallel efficiency of the four parallel approaches for the channel flow problem on the system detailed in Section 2.4. The vertical dotted black lines indicates the practical limit imposed by the modal and elemental discretsations in isolation. The solid purple line shows ideal efficiency, relative to the 16-core case using modal parallelism with iterative elliptic solver.

is less than the ideal case in general, but still reduces runtime significantly as the number of processes increase. In particular, for 512 processes, the performance approaches the ideal case.

Figures 9 and 10 show the parallel efficiency of the various parallelisations of the pipe and channel problem, respectively, normalised against the 16-core modal case with iterative solver. Efficiency of both modal and elemental parallelism reduces with increasing core counts, however, the use of the hybrid approach recovers a significant portion of the lost efficiency. For the pipe the combined approach enables 80% of ideal parallel efficiency to be attained on 512 cores, while for the channel there is an order of magnitude increase in efficiency at 256 cores, compared to the elemental approach.

4.3. Model Calibration

Calibration is the process whereby we identify values for the machine-specific constants a, b and c in Eq. (3). To simplify the calibration process, we combine the costs of the elliptic solves and split the timings into those for computation and those for communication as

$$T^{NS} = a_1 \cdot T^A_O + a_2 \cdot T^A_C + b_1 \cdot T^E_O + b_2 \cdot T^E_C.$$

To illustrate the use of the model, six measurements were taken of the time taken to solve the pipe problem using the elemental and Fourier parallel decomposition. The model T^{NS} was implemented in MATLAB and the required coefficients were calculated using a least-squares algorithm as

$$a_1 = 0.45 \cdot 10^{-6} \quad a_2 = 0.2 \quad b_1 = 3.15 \cdot 10^{-6}$$

and

$$b_2 = \begin{cases} 400, & \text{if } N^{plane}_{el} / \boldsymbol{P}_{XY} < 4, \\ 10, & \text{otherwise.} \end{cases}$$

The coefficient b_2 is multi-valued since performance of the elemental parallel decomposition typically degrades sharply when

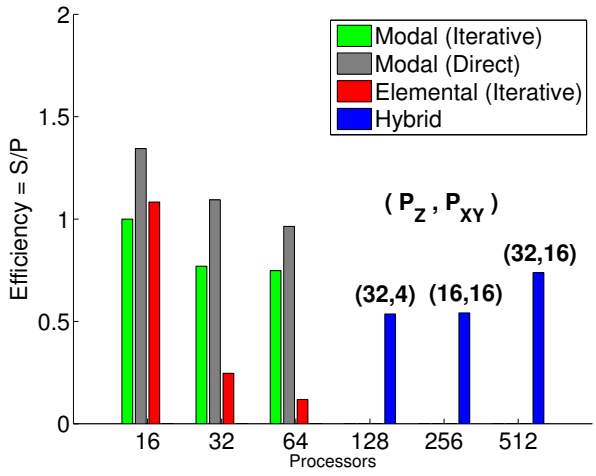

Figure 9: Turbulent pipe flow parallel simulation - efficiency of parallelisation approaches on a cluster of 8-core nodes. The histograms show the efficiency E of different parallel simulations defined as $E = S/P$ where S is the speed-up and P is the total number of processors used for the simulation. The speed-up is based on the 16-core (2 nodes) run using the Modal (iterative) approach.

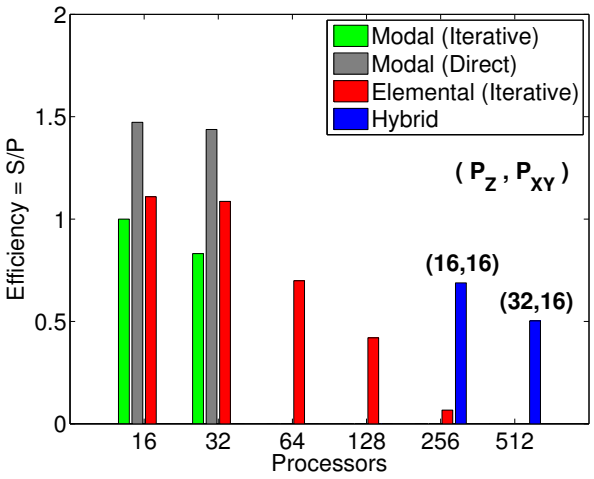

Figure 10: Turbulent channel flow parallel simulation - efficiency of parallelisation approaches on a cluster of 8-core nodes. The histograms show the efficiency E of different parallel simulations defined as $E = S/P$ where S is the speed-up and P is the total number of processors used for the simulation. The speed-up is based on the 16-core (2 nodes) run using the Modal (iterative) approach.

there are fewer than four elements per process, often due to imbalance in the mesh distribution amongst the processes.

4.4. Model validation

To quantify the accuracy of predictions using the calibrated performance model in the hybrid regime, we apply the model to the turbulent pipe flow example. Results presented are obtained by averaging 1000 measurements of the timings for each of the components of the time-stepping algorithm. The choice of elliptic solver has a significant impact on performance, and as in the modal parallelism case, the timings for the elliptic solves are measured using both a direct method using LAPACK and an iterative conjugate gradient approach. The four parallelisation types used throughout the remainder of this section are therefore modal (iterative and direct), elemental (iterative) and hybrid (iterative). Strong scaling is performed for the different parallelism strategies and results are normalised by the timings for the 16-core modal approach using the iterative solver.

The model, outlined in Section 3, is calibrated for the turbulent pipe flow example using simulations executed using both modal parallelism and elemental parallelism in isolation. These timings are consequently accurately reproduced by the model, as shown by the dashed orange lines in Figure 7. The solid orange line in the same figure shows predicted runtimes using the performance model in the hybrid regime. Good agreement is observed and the model correctly identifies the best performing hybrid case, timings of which are shown by the blue triangles.

5. Discussion

In this paper we presented a technique to parallelise a 3D incompressible Navier-Stokes algorithm discretised using a 2D spectral/*hp* element mesh coupled with a Fourier expansion in a third geometrically homogeneous direction. The implementation enables a flexible mixture of both elemental parallelism and modal parallelism. We have illustrated the hybrid parallelism technique on two prototype problems: turbulent flow in an axisymmetric pipe and turbulent flow in a channel. Both problems enjoy increased parallelism through the approach and consequently improved runtimes and greater energy efficiency. The optimal weighting of these strategies can be systematically chosen through the construction of a performance model, calibrated to a specific system through measurements of the two parallelism approaches independently. This enables rapid selection of the highest-performing combination of the strategies without costly trial-and-error testing.

In the modern HPC environment, where energy is of increasing concern, selecting the optimal implementation to maximise performance is becoming increasingly important. Although experience and intuition can generally suggest the most suitable parallelisation approach for a specific problem, the decision is in general highly challenging, particularly when moving between HPC systems or when tackling problems on a range of different domains with the same algorithm. Even from a purely theoretical perspective, it can be appreciated that a single parallel approach can not be optimal in all situations and this has been confirmed through numerical experimentation.

The number of degrees of freedom in the $xy-$plane and the number of Fourier modes are the first indicators of which technique is more appropriate. We recognise that problems with a high number of modes compared to the number of elements per plane appear to benefit from the modal parallelisation approach. On the other hand, a domain discretisation with a larger number of elements than Fourier modes generally benefits from an elemental decomposition technique. However, the relative performance of different approaches cannot be determined en-

tirely through operation counts. Accounting for specific hardware characteristics and the latency and bandwidth available on the communication pattern is essential to accurately predict the optimal strategy. Parallelisation approaches requiring a large number of messages, such as the mesh-decomposition parallelisation, can suffer from poor performance if the interconnect latency is high. These types of parallel techniques are therefore efficient on shared memory machines or low-latency interconnects.

In general we wish to be able to tackle a range of problems where those quantities can vary, potentially reaching extreme values. It is therefore clearly beneficial to have both parallelisation strategies available within a single codebase and this study illustrates the advantages of being able to combine them in a flexible manner to achieve lower runtimes on a fixed number of processors. A further benefit is the extension of strong scalability possible through the use of the hybrid parallel implementation. The potential of new machines are often exploited by investigating even larger problems than previously explored and in finer detail, or using larger Reynolds numbers, and to capitalise on weak scalability. Good weak scalability generally follows from good strong scalability and by increasing an algorithms strong scalability, simulations can be run faster and on larger machines.

It should be noted that not all the hybrid parallel approaches we tested provided good performance. Depending on the mesh topology, partitioning and the number of Fourier modes on each processor, some strategies may not perform efficiently. We note, for example, that the optimal 128-core hybrid parallel case for the turbulent pipe in Figure 7 has a similar runtime to the modal parallelism using the direct solver with only 64 cores. Conversely, we showed that certain choices may result in reduced computational time without increasing the number of processors. This is the case of the modal approach when using a direct solver, which generally performs as well as an elemental parallelisation method with twice the number of CPUs. Minimising the energy consumption when running a simulation is a point of interest in current high performance computing research. Implementation flexibility plays an important role in addressing these goals.

In terms of limitations, we have disregarded some pieces of the algorithm in order to focus on the two main routines, namely the advection and the elliptic operators. This is a typical approach when creating a scalability model [20], although it may introduce some errors. The calibration has been carried out by monitoring the solution time on the SGI Altix ICE 8200 EX system, therefore the coefficients presented here must be considered specific for that machine. Finally, the model, and therefore the results presented in this paper, are specific to *Nektar++*. However, it still provides valuable insight and can suggest overall guidelines on typical choices of parallelisation strategy on large HPC resources.

6. Acknowledgments

AB and SJS acknowledge support from EPSRC under grant EP/H000208/1. CDC acknowledges support of the British Heart Foundation under grant FS/11/22/28745. DS acknowledges support from CNPq and FAPESP. DM acknowledges support under the Laminar Flow Control Centre funded by Airbus/EADS and EPSRC under grant EP/I037946.

References

[1] N. Kroll, C. Hirsch, F. Bassi, C. Johnston, K. Hillewaert, IDIHOM: Industrialization of High-Order Methods-A Top-Down Approach: Results of a Collaborative Research Project Funded by the European Union, 2010-2014, Vol. 128, Springer, 2015.

[2] S. Pope, Turbulent Flows, Cambridge University Press, 2000.

[3] H. Meuer, E. Strohmaier, J. Dongarra, H. Simon, Top500 supercomputer sites, http://www.top500.org (2013).

[4] G. Karniadakis, S. Sherwin, Spectral/hp element methods for computational fluid dynamics, 2nd Edition, Numer. Math. Sci. Comp., Oxford University Press, Oxford, 2005.

[5] C. Cantwell, S. Sherwin, R. Kirby, P. Kelly, From h to p efficiently: selecting the optimal spectral/hp discretisation in three dimensions, Mathematical Modelling of Natural Phenomena 6 (3) (2011) 84–96.

[6] C. Cantwell, S. Sherwin, R. Kirby, P. Kelly, From h to p efficiently: Strategy selection for operator evaluation on hexahedral and tetrahedral elements, Computers & Fluids 43 (1) (2011) 23–28.

[7] P. Vos, S. Sherwin, R. Kirby, From h to p efficiently: Implementing finite and spectral/hp element methods to achieve optimal performance for low- and high-order discretisations, Journal of Computational Physics 229 (13) (2010) 5161–5181.

[8] H. Blackburn, D. Barkley, S. J. Sherwin, Convective instability and transient growth in flow over a backward-facing step, Journal of Fluid Mechanics 603 (2008) 271–304.

[9] C. Cantwell, D. Barkley, H. Blackburn, Transient growth analysis of flow through a sudden expansion in a circular pipe, Physics of Fluids (1994-present) 22 (3) (2010) 034101.

[10] K. Avila, D. Moxey, A. de Lozar, M. Avila, D. Barkley, B. Hof, The onset of turbulence in pipe flow, Science 333 (6039) (2011) 192–196.

[11] G. Karniadakis, Spectral Element-Fourier Methods for Incompressible Turbulent Flows, Computer Methods in Applied Mechanics and Engineering 80 (1990) 367–380.

[12] A. Grama, A. Gupta, V. Kumar, Isoefficiency: measuring the scalability of parallel algorithms and architectures, IEEE Parallel Distributed Technology Systems Applications 1 (3) (1993) 12–21.

[13] C. Crawford, C. Evangelinos, D. Newman, G. Karniadakis, Parallel benchmarks of turbulence in complex geometries, Computers & Fluids 25 (1996) 677.

[14] C. Evangelinos, G. Karniadakis, Communication Performance Models in Prism: A Spectral Element-Fourier Parallel Navier-Stokes Solver, in: Proceedings of the 1996 ACM/IEEE Conference on Supercomputing (SC'96), 1996.

[15] P. Fischer, Analysis and Application of a Parallel Spectral Element Method for the Solution of the Navier-Stokes Equations, Computer Methods in Applied Mechanics and Engineering 80 (1990) 483–491.

[16] P. Fischer, E. Rønquist, Spectral element methods for large scale parallel Navier-Stokes calculations, Computer Methods in Applied Mechanics and Engineering 116 (1–4) (1994) 69–76.

[17] P. Fischer, Parallel domain decomposition for incompressible fluid dynamics, Contemporary Mathematics 157 AMS (1994) 313–322.

[18] P. Fischer, A. Patera, Parallel Simulation of Viscous Incompressible Flows, Ann. Rev. Fluid Mech. 26 (1994) 483–528.

[19] P. Fischer, An overlapping Schwarz method for spectral element solution of the incompressible Navier-Stokes equations, J. Comput. Phys. 133 (1997) 84–101.

[20] C. Hamman, R. Kirby, M. Berzin, Parallelization and scalability of a spectral element channel flow solver for incompressible Navier–Stokes equations, Concurrency and Computation: Practice and Experience (March) (2007) 1–7.

[21] S. Orszag, Spectral methods for problems in complex geometries, J. Comput. Phys. 37 (1) (1980) 70–92.

[22] G. Karniadakis, M. Israeli, S. Orszag, High-Order Splitting Methods for the Incompressible Navier-Stokes Equations 414443 (1991) 414–443.

[23] E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, D. Sorensen, LAPACK Users' Guide, 3rd Edition, Society for Industrial and Applied Mathematics, Philadelphia, PA, 1999.

[24] G. Karypis, METIS's Manual, Department of Computer Science and Engineering, University of Minnesota, Minneapolis, MN 55455, 5th Edition (March 2013).

[25] P. Fischer, J. Lottes, D. Pointer, A. Siegel, Petascale algorithms for reactor hydrodynamics, Journal of Physics: Conference Series 125.

[26] R. Kirby, S. Sherwin, The Nektar++ project, http://www.nektar.info (2006).

[27] C. D. Cantwell, D. Moxey, A. Comerford, A. Bolis, G. Rocco, G. Mengaldo, D. de Grazia, S. Yakovlev, J.-E. Lombard, D. Ekelschot, B. Jordi, H. Xu, Y. Mohamied, C. Eskilsson, B. Nelson, P. Vos, C. Biotto, R. M. Kirby, S. J. Sherwin, Nektar++: An open-source spectral/hp element framework, Comput. Phys. Commun. 192 (2015) 205–219. doi:10.1016/j.cpc.2015.02.008.

[28] J. Demmel, M. Heat, H. van der Vorst, Parallel numerical linear algebra, Acta Numerica (1993) 111–197.

[29] H. Koberg, Turbulence control for drag reduction with active deformation, Ph.D. thesis, Imperial College London (University of London) (2007).

[30] J. Kim, P. Moin, R. Moser, Turbulence statistics in fully developed channel flow at low reynolds number, Journal of Fluid Mechanics 177 (1987) 133–166.