Petascale algorithms for reactor hydrodynamics

Paul Fischer, James Lottes, David Pointer and Andrew Siegel

Argonne National Laboratory, Argonne, IL 60439, USA

E-mail: fischer@mcs.anl.gov

Abstract. We describe recent algorithmic developments that have enabled large eddy simulations of reactor flows on up to $P = 65,000$ processors on the IBM BG/P at the Argonne Leadership Computing Facility.

1. Introduction

Petascale computing is expected to play a pivotal role in the design and analysis of next-generation nuclear reactors. Argonne's SHARP project is focused on advanced reactor simulation, with a current emphasis on modeling coupled neutronics and thermal-hydraulics (TH). The TH modeling comprises a hierarchy of computational fluid dynamics approaches ranging from detailed turbulence computations, using DNS (direct numerical simulation) and LES (large eddy simulation), to full core analysis based on RANS (Reynolds-averaged Navier-Stokes) and subchannel models. Our initial study is focused on LES of sodium-cooled fast reactor cores. The aim is to leverage petascale platforms at DOE's Leadership Computing

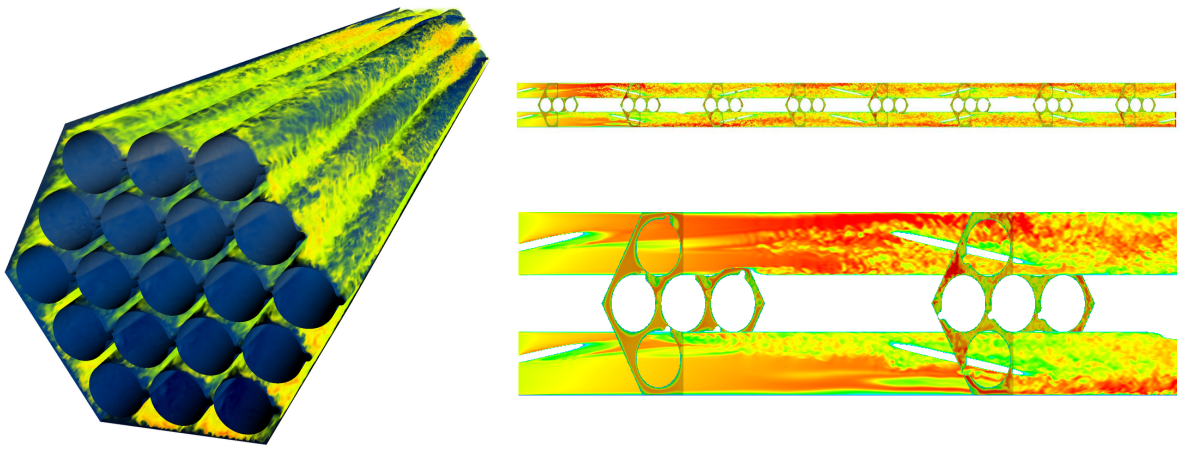

Figure 1. Turbulence in wire-wrapped subassemblies visualized by axial velocity distributions. Periodic boundary conditions applied over a single pitch, $z \in [0, H]$, are used for the 19-pin case (left) while inflow/outflow conditions at $z = 0/3H$ are used for the 7-pin case (right). Transition to turbulence from a uniform inlet flow occurs at $z < H/2$ in the 7-pin case (lower right).

Facilities (LCFs) to provide detailed information about heat transfer within the core and to provide baseline data for less expensive RANS and subchannel models [5].

Figure 1 shows recent simulations of turbulent coolant flow in 19- and 7-pin assemblies. Coolant passes through interior flow channels between the pins and through exterior edge and corner channels between the pins and the walls. The pins are separated by helically wrapped spacer wires that also divert flow between channels and enhance mixing. Questions to be answered by these initial studies include the degree of cross-assembly mixing induced by the spacer wires, the amount of bypass flow in the edge channels, and the magnitude of the pressure drop through the assembly. These fine-scale simulations are providing data and physical insight previously accessible only through experiment.

LES of full or even partial subassemblies is challenging. For pin diameter D, the hydraulic diameter for an interior channel is $D_h \approx 0.4D$. The wire pitch is $H \approx 60D_h$, which is much longer than the correlation length at typical Reynolds numbers, $Re_h := UD_h/\nu \approx 50,000$. The number of channels is roughly twice the number of pins, and the domain length is $\approx 10H$. Thus, for a 217-pin subassembly one is faced with LES in a channel of length equivalent to $\approx 400\times10\times60D_h = 240000D_h$. Fortunately, this figure can be reduced by exploiting symmetries. In particular, the relatively short entrance length ($< 60D_h$; see figure 1) for these flows permits the use of axial periodicity such that one only needs to simulate a single wire pitch. Even with periodicity, the range of scales encountered in a single subassembly has driven the computational requirements for our LES code, Nek5000, to unprecedented levels.

In the remainder of this paper, we describe recent algorithmic advances that are enabling scalable simulations on the DOE's new petascale architectures.

2. Petascale flow solvers

Our large eddy simulations are based on the spectral element code, Nek5000, which has been designed from the outset for distributed-memory platforms. The spectral element method (SEM) combines the geometric flexibility of finite elements with the minimal numerical dispersion and dissipation of spectral methods, making it ideally suited to LES in large complex domains, where one is interested in accurate propagation of the principal energy carrying eddies over long times. In the SEM, the solution within each of E elements is represented as a tensor-product of Nth-order Lagrange polynomials based on the Gauss-Lobatto-Legendre nodal points, resulting in $n \approx EN^3$ degrees of freedom per scalar field. Our current LES formulation relies on slight (resp., 1.25 and 5%) filtering of modes k=N-1 and N, which provides an energy drain at the unresolved grid scale. In well-resolved regions, the action of the filter is void and spectral accuracy is retained.

The principal computational bottleneck in simulating unsteady incompressible and low-Mach number flows is the elliptic problem governing the pressure, which must be computed implicitly at each timestep. For large unstructured problems, the resultant discrete Poisson problem is most efficiently solved using multilevel iterative methods embedded within a Krylov subspace projection scheme such as conjugate gradients or GMRES. For the SEM, we employ variational multigrid using *local* overlapping Schwarz methods for element-based smoothing at resolution N and $\approx N/2$, coupled with a *global* coarse-grid problem based on linear elements. (A detailed description of our solver can be found in [2, 3].)

The coarse-grid problem is a well-known impediment to scalability. In the past, we have used a projection-based scheme to solve the coarse problem $A\underline{x} = \underline{b}$. If A is sparse symmetric positive definite, then there exists a quasi-sparse matrix X with $O(n^\alpha)$ nonzeros, $1 < \alpha < 2$ such that $\underline{x} = XX^T\underline{b}$. For linear finite element discretizations of the 3D Poisson problem, one can compute $\underline{x}$ with a P-processor work complexity of $T_a \approx 10n^{5/3}/P$ and a communication complexity involving only $2\log_2 P$ contention-free messages of length $\leq 5n^{2/3}$ [6]. While this highly parallel and communication-minimal scheme has been quite effective for $(P, n) < (10^4, 10^5)$,

its superlinear costs are ultimately limiting, and we have recently turned to algebraic multigrid (AMG) as an alternative.

Our AMG approach is guided by the two-grid asymptotic convergence rate, equal to the spectral radius of the error propagation matrix $\rho(\ (I-BA)(I-PA_c^{-1}P^TA)\ )$, where the (AMG) coarse operator is $A_c = P^TAP$. Here we see that the iteration is determined by the choice of *smoother*, B, and by the choice of the $n \times n_c$ *prolongation* matrix, P. P is further constrained by specifying that the coarse "C-variables" be a subset of the original variables. Ordering these variables last gives rise to the block forms

$$A = \begin{pmatrix} A_{ff} & A_{fc} \\ A_{cf} & A_{cc} \end{pmatrix}, \qquad P = \begin{pmatrix} W \\ I \end{pmatrix}, \qquad B = \begin{pmatrix} \hat{B}_{ff} & 0 \\ 0 & 0 \end{pmatrix}.$$

The key components of the AMG scheme are the choice of the C-variables, the prolongation weights (W), and the smoother. (We discuss the form of B below.)

We take $\kappa(D_{ff}^{-1/2}A_{ff}D_{ff}^{-1/2})$ to be a measure of coarsening quality, as it determines both the support required for the prolongation weights and the number of smoothing steps required. Our coarsening procedure assigns coarse variables in order to eliminate the maximal Gershgorin disc radii of $D_{ff}^{-1/2}A_{ff}D_{ff}^{-1/2} - I$, where D_{ff} is the diagonal of A_{ff}. The procedure stops once the a priori bound on the quality measure is below some tolerance.

Our prolongation weights are based on the energy-minimizing interpolation of Wan, Chan, and Smith [7], which is the solution to a constrained minimization problem that may be viewed as a tractable proxy to the problem of minimizing the departure of W from the minimal energy weights, $-A_{ff}^{-1}A_{fc}$, given a sparsity pattern on W. We determine the supports in a spirit similar to sparse approximate inverse (SAI) preconditioners by locally approximating the columns of $A_{ff}^{-1}A_{fc}$ and widening the support until the approximation is accurate to a given tolerance. For the smoother, $\hat{B}_{ff}$, we choose diagonal SAI accelerated in a Chebyshev polynomial (typically of degree 2 or 3). The choice of using "F-relaxation" was motivated by the fact that no additional benefit may come from using a full smoother in a two-level asymptotic convergence rate analysis and that, by approximately solving equations governed by A_{ff}, one may legitimately employ Krylov subspace (e.g., Chebyshev) acceleration.

The success of our AMG solver derives from its robust convergence properties (overall convergence is the same as with the direct XX^T solver) and the rapid reduction in the work with each level of the V-cycle. For the 19-pin case of figure 1, which starts with 120 million points, the AMG grid sizes are 417600, 267100, 83346, 37097, 19339, 7912, 2237, 459, 92, 15, 2, and 1. The challenge for AMG is to minimize communication overhead, as we discuss below.

Significant improvements in scalability derived from rewriting our central communication kernel, *gs()*, which implements the interface exchange (gather-scatter) between spectral elements. The *gs()* kernel is a general-purpose utility for implementing parallel matrix-vector products with a particularly lightweight interface. In a setup phase, one invokes *gs_handle = gs_setup(glo_num,*n_p*)*, where *glo_num()* is a local list of n_p integers containing a global identifier (e.g., column number) for each entry. For any repeated entries in *glo_num()* (locally or on other processors) the call *gs(gs_handle,* $\underline{u}$*,'+')*, will return the sum of the corresponding elements of $\underline{u}$. In short, if Q is an arbitrary Boolean matrix, *gs()* implements the symmetric operation QQ^T [1]. The code supports as its last argument any commutative/associative operator as well as a mode for vector fields of arbitrary dimension. The new setup routine discovers interprocessor connections using binning, which is implemented using a crystal-router- (CR-) based all-to-all communication [4]. The CR strategy is simply stated. Any processor $p < P/2$ having data needed by any processor $q \geq P/2$ sends its data to $p + P/2$. Processors in the upper half reciprocate. One then bisects the processor set and recurs, and the exchange terminates after

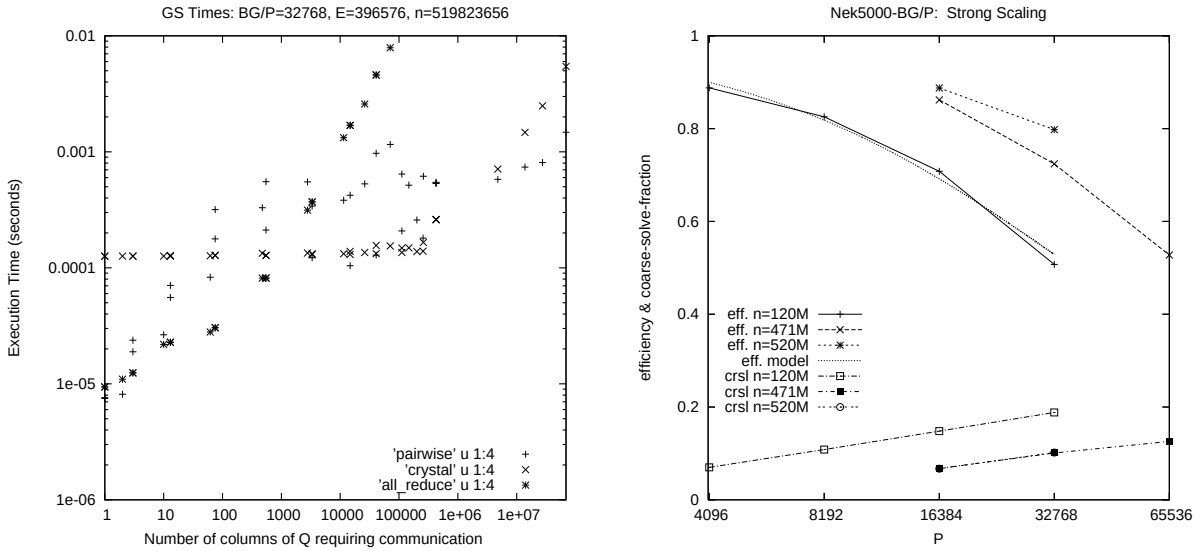

Figure 2. Nek5000 performance on BGP: (left) per-call communication costs for several implementations of the *gs()* kernel versus number of nontrivial columns for the case $P = 32768$; (right) efficiency and coarse solve fraction for problems with n=120M, 470M, and 520M gridpoints.

$\log_2 P$ phases. With this strategy, *gs_setup()* requires $\approx .09$ seconds for n=150 million on 32768 processors of BG/L. (We remark that the stand-alone *gs()* code is available from the authors.)

While robust, general, and easy to use, *gs()* was originally designed for *nearest-neighbor* communications—effectively, Q sparse—and relied exclusively on pairwise processor exchanges to swap edge and face data. AMG, which also requires general unstructured communication, can have stencil widths of several hundred at the intermediate levels. A pairwise exchange strategy in this case would require hundreds of messages and suffer significant latency. To this end, we have restructured *gs()* to support several exchange strategies within the same interface and with the chosen strategy determined by a test that is invoked automatically at setup. In addition to pairwise exchanges, *gs()* now supports *all_reduce()*- and CR-based exchanges. The former approach simply puts the entries from each processor onto a null-vector of length n and then condenses the result using the native MPI *all_reduce()*. This is very fast on BG/P, which has a dedicated tree network for this task. The latter approach employs the CR scheme described above, with condensation taking place whenever shared values collide at intermediate stages. Figure 2, left, shows the performance envelope for the three strategies as a function of the number of columns of Q having more than one entry, taken over all calls to *gs_setup()* for the 7-pin configuration of Fig. 1 with N=11. While the plot does not indicate the (important) multiplicity of each column of Q, it nonetheless gives some insight to the communication patterns. In particular, *pairwise* is generally fastest for the large sparse Qs, while *all_reduce()* is superior for the smallest systems and *CR* is optimal in the middle range.

3. Parallel scaling on BG/P

Strong scaling studies were performed for models shown in figure 1. Case 1 is the 19-pin model with E=359424 and N=7, corresponding to n=120 million points. Case 2 is the same model, but with N=11 (n=471 million) and Case 3 is the 7-pin $\times$ 3-pitch inflow/outflow configuration, with $E = 396576$ and N=11 (n=520 million). Each case is run for 100 timesteps without I/O.

Table 1 gives a detailed breakdown of the parallel overhead for Case 1. Reported times are:

Table 1. BG/P solution times (seconds) for 19-pin case, $N = 7$.

Case	Total	QQ^T	Coarse	all_reduce()
x4096	1994	125	1180	1.2
a4096	1112	125	192	1.4
b4096	846	126	25	1.
8192	460	88	22	1.
16384	266	64	20	1.

total; nearest-neighbor (QQ^T) exchanges for the SEM, including the $N/2$ multigrid exchanges; coarse-grid solve; and all vector reductions. The entry *x4096* uses the XX^T solver, while *a4096* uses a variant of the AMG solver which selects between the pairwise *gs()* exchange or *all_reduce()* to optimize the AMG communication. The entries *b4096* and below are for the current scheme, which enters a third communication option in the mix, namely, the *crystal_router()* routine.

Parallel efficiencies (figure 2, right) are derived from from the solution-time model, $T(P) = W_s + W_p/P$. For any value of P, given $T(P)$ and $T(P/2)$, one can estimate the serial-to-parallel work ratio $W_{sp} := W_s/W_p$, from which the efficiency is $\eta_p = (1 + W_{sp})/(1 + PW_{sp})$. For Case 1, W_{sp} ≈2.72e-5, based on an average over runs on 1, 2, 4, and 8 racks, with 4096 cores/rack. For Case 2, W_{sp} ≈1.17e-5, based on 4, 8, and 16 racks. For Case 3, W_{sp} ≈7.74e-6, based on 4 and 8 racks. At P=32768, Case 1 achieves roughly 50% efficiency, which is in fact reasonable given that n/P is only ≈ 3700 in this case. Case 2 sustains roughly 50% efficiency at P=65536 with $n/P \approx 7300$ and Case 3 realizes 80% efficiency for P=32768 with $n/P \approx 15870$. The fraction of time spent in the coarse grid is $< 20\%$ for all cases.

4. Conclusion

We have presented algorithmic and performance considerations for petascale simulations of incompressible flows. Computations on the ALCF BG/P are already yielding important results in the analysis of reactor core flows, including establishment of core entry lengths.

Acknowledgments

This work was supported by the U.S. Dept. of Energy under Contract DE-AC02-06CH11357. Computer time on the Argonne Leadership Computing Facility was provided through a 2008 DOE Office of Science INCITE Award. We thank Hank Childs and David Bremer from the Lawrence Livermore National Laboratory VisIt group for assistance with data analysis.

References

[1] Deville M, Fischer P and Mund E 2002 High-order methods for incompressible fluid flow (Cambridge University Press)

[2] Fischer P and Lottes J 2004 In R Kornhuber, R Hoppe, J Priaux, O Pironneau, O Widlund and J Xu eds *Domain Decomposition Methods in Science and Engneering* (Springer: Berlin)

[3] Fischer P, Loth F, Lee S, Smith D and Bassiouny H 2007 *Comput. Methods Appl. Mech. Engrg.* **196** 3049–3060

[4] Fox G C, Johnson M A, Lyzenga G A, Otto S W, Salmon J K and Walker D W 1988 *Solving Problems on Concurrent Processors* (Prentice-Hall: Englewood Cliffs, NJ)

[5] Pointer W, Fischer P, Siegel A and Smith J 2008 *Proc. Int. Congress on Advances in Nuclear Power Plants* (Anaheim, CA) Paper 8252

[6] Tufo H and Fischer P 2001 *J. Parallel Distrib. Comput.* **61**:151–177

[7] Wan W L, Chan T F and Smith B 1999 *SIAM J. on Sci. Comp.* **21(4)**:1632–1649