IESP Whitepaper: PDE-based applications and solvers at extreme scale

David Keyes
Columbia University & SciDAC TOPS project

The thirst for extreme floating-point processing rates is unquenchable in the foreseeable future, being driven by the need for: (1) better resolving the full ranges of length or time scales in multiscale phenomena, (2) accommodating physical effects with greater fidelity, (3) allowing the model degrees of freedom in all relevant dimensions, (4) better isolating artificial boundary conditions in PDE models and better approaching realistic levels of dilution in particle models, (5) optimizing or controlling physical scenarios (by solving inverse problems) once they are adequately resolved by forward models, (6) quantifying uncertainty, and (7) improving statistical estimates. As applications stretch to take full advantage of extreme architectures, however, the computational complexity of some algorithms, such as Courant-stability-limited explicit solvers as well as some linear and nonlinear solvers, grows superlinearly in memory size, making it impossible to weak scale, even though memory capacity would seem to allow it. Extreme scales put a premium on finding "optimal" algorithms, whose complexity is at worst log-linear in problem size; any suboptimal component will ultimately dominate the execution profile. In fact, to justify the acquisition and operating costs of exascale hardware, one needs to be concerned not only with complexity exponents, but also with the coefficients in front of the power laws, which can vary considerably from one formulation to another. *The availability of high capability architecture makes algorithms more, not less, important.*

Fortunately, algorithms such as linear solvers have kept pace with extreme scales, and optimal versions are known for many PDE-based formulations of driving applications. Therefore modelers who can cast their simulations in terms of these formulations (*e.g.*, sequences of Poisson solves to build up a preconditioner for a multicomponent system of more general type) may weak scale to 10^5 processor cores today, on a massively parallel computer with a log-diameter network. The logarithm, if it does not also arise from other causes, is a consequence of the global reduction operations that are present in Newton, Krylov, and other algorithms and *ultimately* degrades the marginal effectiveness of additional processor-memory elements if the synchronization stranglehold is not deferred by reducing its frequency. Furthermore, the marginal effectiveness of additional processors dividing the bandwidth of a memory shared among many processors may be nearly zero in many sparse algorithmic kernels.

As a further threat to effective use of extreme scale hardware, we note that progressive, mathematically beneficial trends in algorithms, such as increased use of unstructured meshes and adaptive discretizations that yield more accuracy per degree of freedom stored or flop performed at the expense of increased indirection, more conditionals, or more integer operations per flop, inveigh against the uniformity and predictability that are required to obtain maximum use of the floating point hardware. Traditional performance metrics focusing on floating point rates *only* in highly unbalanced hardware have long ceased, in general, to be reliable guides to the merits of a numerical computation. Instead, performance optimizers should hunt for each successive bottleneck – whether bandwidth, latency, number of integer load/store units, or whatever – and ask what

algorithmic alternative could relieve it by exploiting unused capacity in some other hardware resource.

Solvers are just one of many algorithms that must scale. Tools for managing meshes, fields, and particles, *e.g.,* their generation, partitioning, adaptation, interpolation, and for constructing of the discrete equations from the underlying models must all be scalable, as well, or Amdahl's Law will impose a limit to scalability that is asymptotically independent of process granularity. The algorithmic techniques required to support simulations of interest at extreme scales include CAD-to-mesh geometric adaptivity, solution-based adaptivity, mesh partitioning, discretizations of virtually all types (with attention to advanced high-order discretizations), contact-detection algorithms, optimal implicit solvers, stiff method-of-lines integrators, kinetic and particle methods, unconstrained and constrained optimization (for parameter identification, control, design, etc.), sensitivity analysis (statistics- and derivatives-based), and uncertainty quantification. Extreme-scale simulation represents an opportunity for developers of the enabling technologies in applied mathematics and computer science to demonstrate a paradigmatic shift that they have envisioned for years as completely new application codes are written. The connective and control code and the majority of the means of interchange of data between code components will have to be rewritten together with algorithmic kernels take advantage of modern software practices and high-performance architectures. Virtually all large-scale data structures in existing codes will have to be replaced with distributed versions. In simulations at extreme scales, no data structure whose size scales with the system can be relegated to just one processor-memory element or replicated on each. As the software infrastructure is rebuilt, due attention can be given to extensibility, reusability, object orientation, componentization, portability, performance portability and tuning, code self-description and self-monitoring, and the construction of multi-layered interfaces that enforce correct usage.

Beyond these improvements that are occasioned by extreme scales (though valuable at any scale) the synchronizations that are built into most codes as matters of convenience in programming model must be drastically reduced. New algorithms and new programming models must be found that postpone synchronizations as long as possible. One class of trade-offs that is well developed requires more memory and more nearest-neighbor communication, which in turn allow many relaxation sweeps or Krylov steps to be conducted per synchronization. Another class of trade-offs hierarchically decomposes an implicit solve that involves all degrees of freedom globally into a set of infrequently communicating local implicit solves, with frequent synchronization within the local basins only. Such algorithms are known and are in some nonlinear problems actually demonstrably faster than their globally synchronizing counterparts, though they might in general be expected to be slower. However, full exploitation of asynchronous algorithms requires programming scientific applications much like operating systems, with different priorities assigned to different tasks, depending upon whether they are on or off the critical path, and with data-driven associative communication between them. The SPMD bulk synchronous model that is so convenient to understanding large-scale simulations will have to yield to far more general constructs that are less reproducible and likely far more difficult to verify for correctness and to predict for performance.