Interweaving Timelines for Faster Solutions

IN 1965, Intel cofounder Gordon Moore predicted that the number of transistors in an integrated circuit would double every year, enhancing overall processor performance by increasing clock speed—the rate at which the processors can execute instructions. Ten years later, Moore revised his prediction, reducing the rate of doubling to every two years, a trend that—rather astonishingly—held true for decades. However, by the early 2000s, the pace of advancement in clock speed had slowed as chip components approached fundamental limits in size and the upper bounds of energy usage.

The rapid rise in computing power, once made possible by the succession of ever faster, smaller, and more affordable transistors, enabled scientists to develop and run increasingly complex—and computationally demanding—simulations without lengthening the time to solution. As improvements to processor speeds wane, those who design and use such scientific applications have been faced with the specter of plateauing application performance. Fortunately, researchers continue to look beyond hardware innovations, such as chip components, to speed up simulations by finding and refining application- and algorithm-based methods for reducing solution time, most notably through increased use of parallelism. The focus on parallelism is particularly important because increases to computing power today and in the future will occur only through more, not faster, processors.

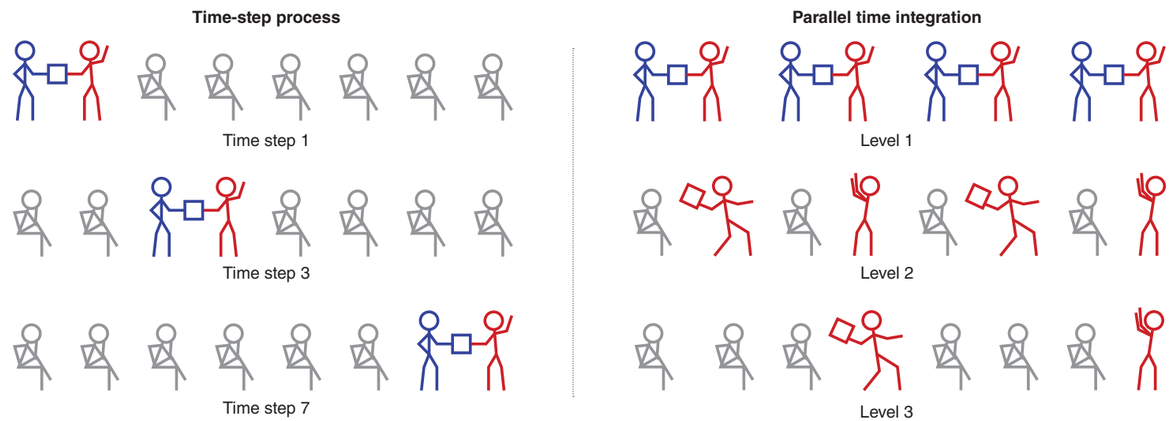
Software Offers Solutions

Parallel processing involves programming a computer system to perform certain tasks simultaneously using multiple processors, rather than sequentially. This approach is essential for efficient operations on today's high-performance computing systems, which might have thousands or even millions of processors. For a typical scientific simulation, intended to help researchers understand how a system evolves through space and time, parallelism might involve simultaneously calculating many grid elements on a spatial grid (spatial parallelism) or many time steps in the simulation (temporal parallelism), or both.

Although spatial parallelism is ubiquitous in scientific computing, parallel time integration is still a nascent area for exploiting performance gains. Researchers began exploring the method back in the 1960s, but it only gained traction in the scientific computing community over the last decade. The reason for this latent interest, surmises Livermore computational mathematician Jacob Schroder, is that the approach is more difficult to execute than more traditional methods, and it seems counterintuitive, since to humans, time is a sequential concept.

Even now, researchers working on parallel time integration face some skepticism, especially from individuals who have not yet exhausted their opportunities for accelerating application performance through more conventional approaches. Computational mathematician Rob Falgout says, "The problem is that many people have not yet faced a bottleneck in their work, and thus they are hard to convince regarding the utility of this method. For them, the issue is down the road, but I find it difficult to imagine that they will not have to address it eventually. Scientists are always striving for greater simulation accuracy." He suggests that since greater accuracy generally requires more computationally expensive

(below left) Typically, time steps in a simulation are solved sequentially, as depicted in this conceptual image that shows figures passing information from one to another in order. (below right) Parallel time integration techniques use the answers from less precise versions of the problem to accelerate the calculation of finer scale versions, which allows the application to more rapidly converge to a solution of the desired accuracy.



simulations, these researchers will ultimately need to exploit every possible avenue for reducing the time to solution for their codes.

At Lawrence Livermore, home to a host of large and sophisticated applications, improvements in application speed and accuracy are driven by national and global security challenges, and thus the need for new approaches is more immediate here than at many institutions. Fortunately, with support from the Laboratory Directed Research and Development Program, Falgout, Schroder, and their Livermore colleagues have developed a promising solution—a novel software that works in tandem with existing high-performance computing applications to calculate all of a problem's time steps simultaneously. Called XBraid, the software has decreased solution time by as much as 50 times for some types of simulations. (See *S&TR*, September 2016, pp. 4–11.)

Scalable and Nonintrusive

As the name suggests, XBraid functions by "braiding" multiple timelines of differing accuracies together for a faster solution, using multigrid methods similar to those the team has successfully applied to speed up spatial calculations. Schroder provides a simple example. If a scientist wants to predict the temperature for a given city with a high level of fidelity—say, by calculating temperature at an interval of once a second for a whole year—and approaches the problem in the standard fashion, the system would have to calculate some 31 million time steps sequentially. By incorporating XBraid, the application would instead simultaneously calculate the solution at several different levels of detail—predicting the temperature, for instance, once a day, once an hour, once a minute, and once a second.

In this example, the only timeline computed sequentially is the coarsest (least precise) one, which has 365 time steps. Those coarse-grained solutions are then fed into the even finer scale problem (the once-an-hour calculation). These solutions are in turn fed into the finer scale version of that problem, and so on, accelerating the solution process at the finest scale. Despite the unconventional approach, the accuracy of the results is the same, up to a user-defined tolerance. Schroder says, "The goal of parallel in time is not to compute a different solution than sequential calculation methods—it is simply to do it more quickly." Further, the method is scalable, so the problem can be sped up by increasing the number of processors working on the calculation.

Although XBraid is neither the first nor the sole successful method of solving time intervals concurrently, its nonintrusive nature provides a huge advantage to those who steward and use today's large scientific applications. XBraid was created with Livermore's current stable of applications in mind, allowing the applications to take advantage of parallel time integration without having to be rewritten, which for a large application could easily take a dozen or more software developers 5 to 10 years to complete.

Power Grids and Neural Networks

Over the past few years, the XBraid team has experimented with optimizing XBraid for various problems, many of which have never successfully incorporated such methods. For one project, the team added temporal parallelism to GridDyn, a Livermore-developed, open-source tool for simulating the electrical power grid. These simulations offer insight into how scheduled interruptions (maintenance, for instance) and unscheduled interruptions (weather, equipment malfunctions, or even an act of terrorism) might affect the supply and distribution of electricity to homes and businesses. The XBraid team focused their initial efforts on simulations involving scheduled outages of grid components. Running GridDyn on Livermore's Quartz cluster, the team demonstrated a roughly 50-fold speed-up for a power system modeling the Western United States.

"Power grid simulations are one of the more promising application areas for parallel time integration," says Schroder, noting that rapid and accurate simulation results are especially important as more renewable power is incorporated into the grid. "Researchers have a weather model to determine wind speed at different times of day. However, one cannot predict precisely when the wind will die," he adds. "If an unexpected disruption occurs, such as if wind power drops sooner than expected, it creates a sudden cascade of events within the network. Fortunately, we are starting to see some success with simulating unexpected disruptions."

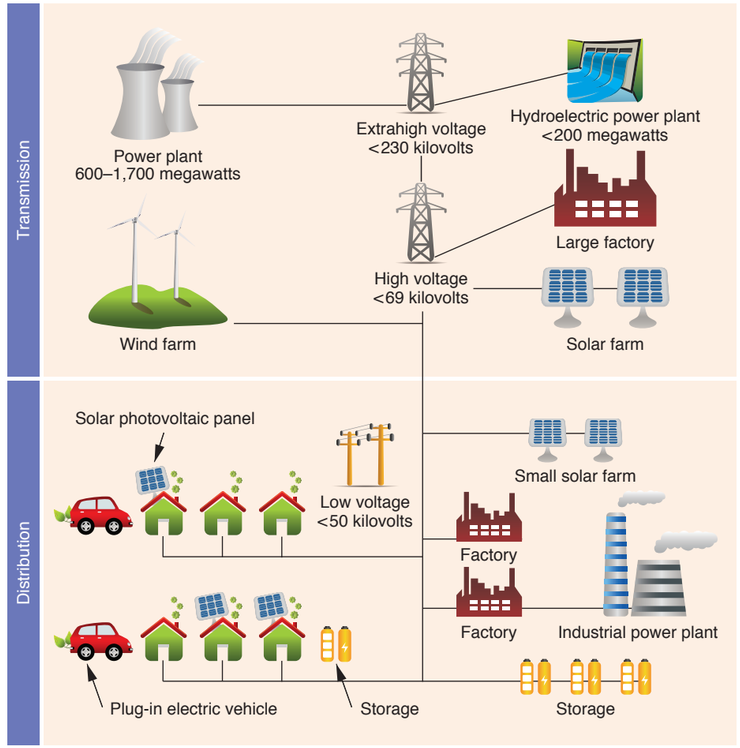
XBraid has also produced favorable results in other challenging areas, such as training artificial neural networks. (See *S&TR,* June 2016, pp. 16–19.) With this popular and effective method of machine learning, data (often in the form of digitized images) are fed through a network, and the resulting output is compared to the desired target, which typically classifies the data into a category. Errors between the output and the target are "back propagated" through the network, assigning blame to the parts of the network responsible for the error. As training proceeds, links in the network strengthen and weaken themselves and converge toward a configuration that minimizes overall error. These networks "learn" by sequentially processing thousands or millions of such training runs.

The idea to apply XBraid to machine learning came one day when Schroder realized the inherent similarities between the serial processing of information in neural network training and that of more traditional time-dependent simulations. By treating training runs as time steps, the team has successfully applied the method such that a 50-training-run problem can be used to help solve a 100-training-run problem, and so forth. In initial feasibility studies, the team saw a six-fold improvement in training time over 13,000 training runs. The work is still in its early stages, but Schroder is optimistic. He says, "Parallel time integration can be highly valuable for machine learning because it provides a new perspective and novel parallel capabilities to the development community."

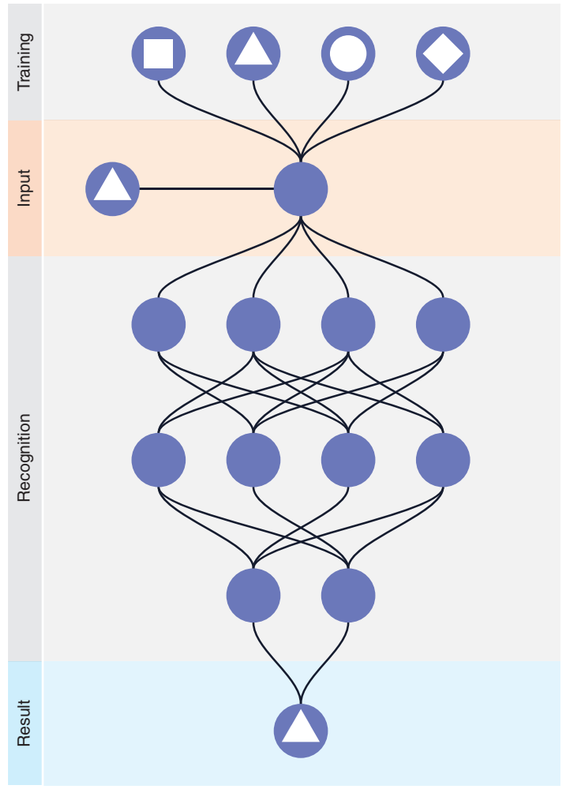
A Necessary Step

XBraid provides a nonintrusive, powerful, open-source solution to the bottleneck posed by performing sequential time steps for problems involving thousands or millions of time steps. Thus far, the XBraid team has demonstrated the viability of this relatively unexplored way to reduce time to solution. XBraid speeds up computations and helps applications make better use of today's supercomputers, for which the number of processors grows with every generation. Asserts Falgout, "Our work is not just an interesting project—it is a necessary one."

—Rose Hansen



This conceptual drawing illustrates the vast, complex resources that may exist in the electrical grid of the future. Pairing the XBraid algorithm with Livermore's GridDyn simulation software could enable faster yet accurate simulations of grid operations, which could benefit both operations and contingency planning.



Sequential training for artificial neural networks often involves showing the network a series of labeled digital images to "teach" it to categorize data correctly. Livermore researchers have sped up this popular machine-learning method by exploiting its similarities with sequential time-step techniques used in more standard scientific simulations.

Key Words: algorithm, artificial neural network, GridDyn, machine learning, Moore's law, multigrid, parallelism, parallel time integration, power grid, XBraid.

For further information contact Rob Falgout (925) 422-4377 (falgout2@llnl.gov).