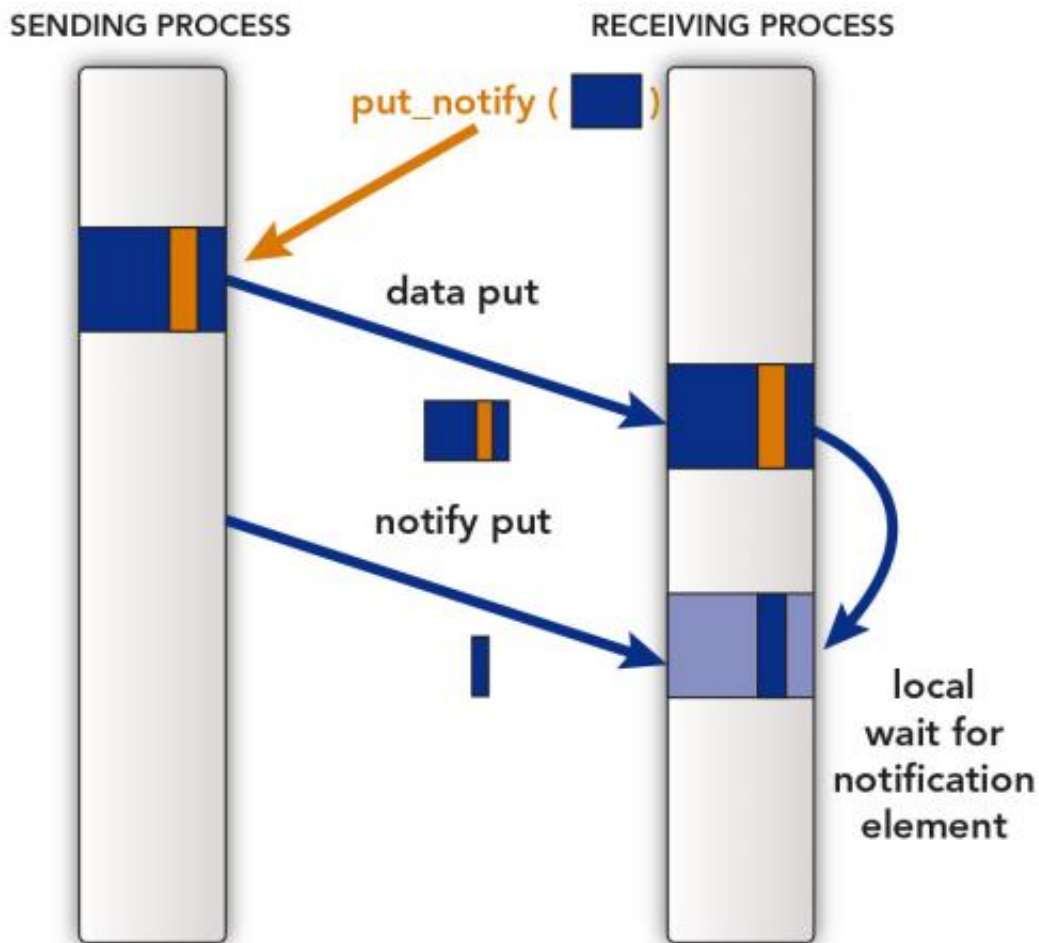# No downtime for communication: New framework allows for asynchronous communication in exascale machines

February 28 2013

Schematic illustration of push data using a put_notify call in Global Arrays.

(Phys.org)—The productivity of a group of colleagues on a project is always more efficient when required information is sent as soon as it becomes available, rather than sending a request for information when it's needed. In the same way, computer algorithms that send data from one process to another when the data becomes available will be more efficient than one that is requested when it will be used. To facilitate designing such algorithms within the Global Arrays programming model framework, DOE researchers at Pacific Northwest National Laboratory designed a new put_notify capability that allows a process to initiate and complete data to another process without synchronization. The novel feature is a notify element that the receiving process can use to asynchronously determine the completion of the data transfer.

To take advantage of enormous resources that next-generation super computers are expected to have, scientific codes must adapt. For example, molecular dynamics (MD) simulation has evolved into a highly useful method for understanding and designing molecular systems. Sophisticated MD analyses can help scientists better understand biomolecular processes, such as protein dynamics and enzymatic reactions. The advantage of using the Global Arrays programming model is that the data transfer can take place while a receiving process is still working on other tasks, a mechanism often referred to as hiding communication behind computation.

To support the needed asynchronous communication and coordination for MD algorithms, the team designed (in Global Arrays, the library-based Partitioned Global Address Space programming model) and implemented the non-blocking put_notify capability. To do this, a two-stage process was created-a put message and notification element-for data communication using a push-data instead of a pull-data model. The researchers were able to show there was discernible time spent between a process sending data and another that receives data. This design reduces the communication bottleneck and the associated load imbalance.

Using novel data-centric capabilities provides unique opportunities to address primary challenges for parallel scalability of MD time-stepping algorithms. In future work, the algorithm will be expanded to include dynamic load-balancing through topology-aware assignment and periodic redistribution of tasks.

Provided by Pacific Northwest National Laboratory