

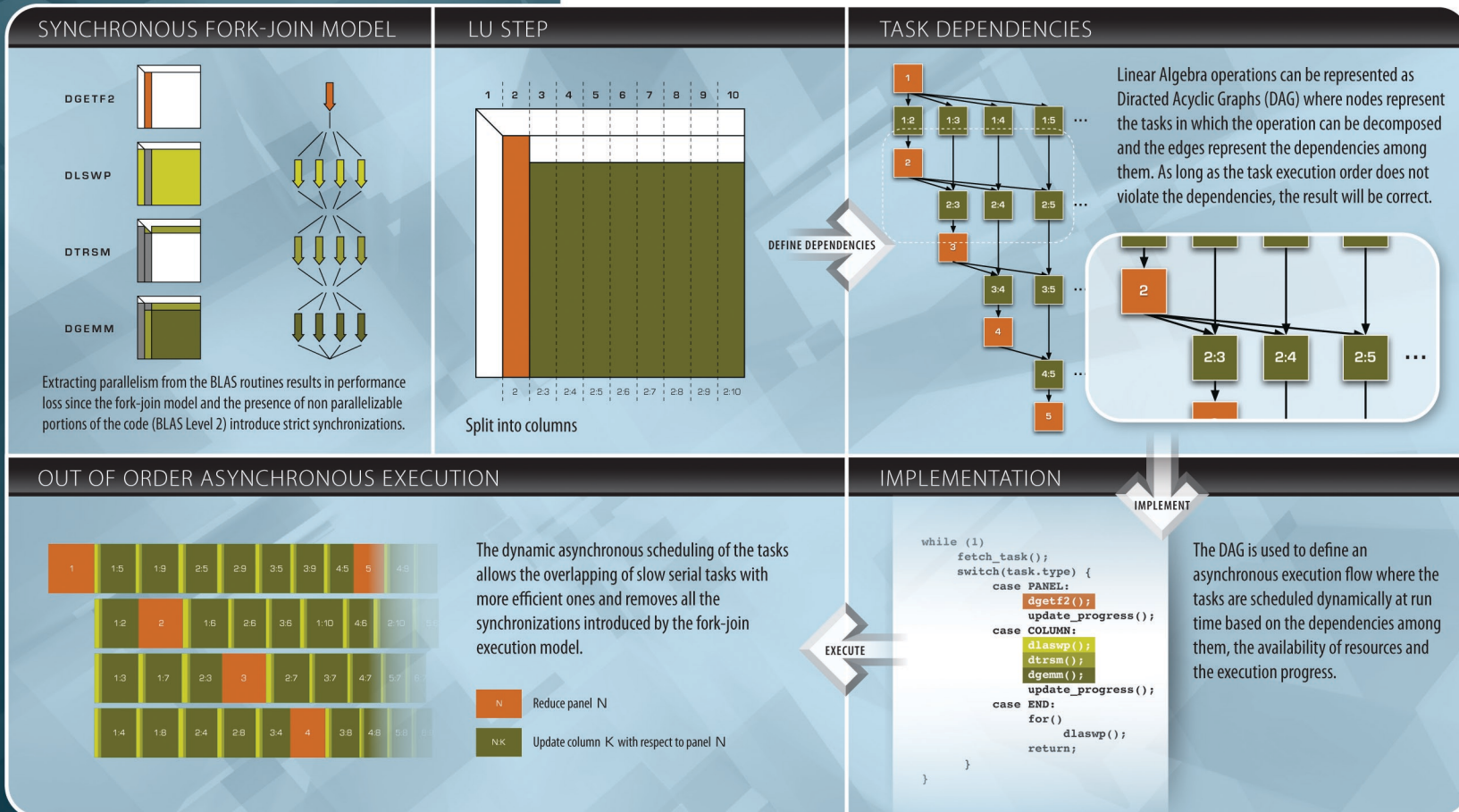
PLASMA

PARALLEL LINEAR ALGEBRA FOR SCALABLE MULTI-CORE ARCHITECTURES

The Parallel Linear Algebra for Scalable Multi-core Architectures (PLASMA) project aims to address the critical and highly disruptive situation that is being faced by the Linear Algebra and High Performance Computing communities due to the introduction of multi-core architectures.

PLASMA's ultimate goal is to create software frameworks that enable programmers to simplify the process of developing applications that can achieve both high performance and portability across a range of new architectures.

The development of programming models that enforce asynchronous out of order scheduling of operations is the concept used as the basis for the definition of a scalable yet highly efficient software framework for Computational Linear Algebra applications.



INNOVATIVE COMPUTING LABORATORY

THE UNIVERSITY of TENNESSEE
 Computer Science Department

MISSION SPONSORS



PLASMA

PARALLEL LINEAR ALGEBRA FOR SCALABLE MULTI-CORE ARCHITECTURES

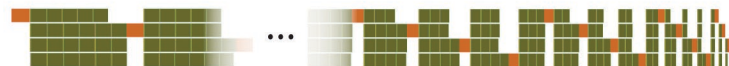
PLASMA's ultimate goal is to create software frameworks that enable programmers to simplify the process of developing applications that can achieve both high performance and portability across a range of new architectures. The development of programming models that enforce asynchronous out of order scheduling of operations is the concept used as the basis towards the definition of a scalable yet highly efficient software framework for Computational Linear Algebra applications.

It is difficult to overestimate the magnitude of the discontinuity that the high performance computing (HPC) community is about to experience because of the emergence of next generation of multi-core and heterogeneous processor designs. For at least two decades, HPC programmers have taken it for granted that each successive generation of microprocessors would, either immediately or after minor adjustments, make their old software run substantially faster. But three main factors are converging to bring this "free ride" to an end. First, system builders have encountered intractable physical barriers - too much heat, too much power consumption, and too much leaking voltage - for further increases in clock speeds. Second, physical limits on the number and bandwidth of pins on a single chip means that the gap between processor performance and memory performance, which was already bad, will get increasingly worse. Finally, the design trade-offs being made to address the previous two factors will render commodity processors, absent any further augmentation, inadequate for the purposes of tera- and petascale systems for advanced applications. This daunting combination of obstacles has forced the designers of new multi-core and hybrid systems, who are searching for more computing power, to explore architectures that software built on the old model are unable to effectively exploit without radical modification. Currently available Linear Algebra software packages rely on parallel implementations of the Basic Linear Algebra Subroutines (BLAS) to take advantage of multiple execution units. This solution is characterized by a fork-join model of parallel execution, which may result in suboptimal performance on current and future generations of multi-core processors since it introduces strict dependencies due to the presence of non parallelizable portions of code. The Parallel Linear Algebra for Scalable Multicore Architectures (PLASMA) project aims to overcome the shortcomings of this approach by introducing a pipelined model of parallel execution.

RESULTS

The execution flow of the sequential implementation with multithreaded BLAS shows how all but one thread are stalled when non-parallelizable tasks are executed. Fixed lookahead can hide the execution of sequential tasks but introduces stalls due to synchronization issues. With dynamic scheduling, sequential tasks are not issued if they introduce stalls in the execution flow.

Threaded BLAS Execution Flow



Fixed Lookahead



Dynamic Scheduling



Performance results on a dual Intel Woodcrest machine (four cores total) for the QR, LU and Cholesky factorizations. The graphs report the comparison between the sequential implementation of the algorithms with multithreaded BLAS and the parallel implementations with dynamic scheduling.

