INNOVATIVE COMPUTING LABORATORY

QUARK

QUEUING AND RUNTIME FOR KERNELS

Productive, Scalable and Efficient Task Execution for Multicore Architectures

QUARK provides a runtime environment for superscalar task execution on shared-memory multicore architectures. It is designed to be **highly productive** with an **easy-to-use API** for **serial task insertion**. QUARK shields the user from the complexities of multicore programming. It determines the data dependencies between the tasks and schedules the tasks to the available threads and resources in a **scalable**, **efficient**, resource-aware and data-aware manner.

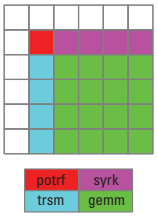
- QUARK provides a library that enables the dynamic superscalar execution of tasks in a multi-core, multi-socket, shared-memory environment.
- QUARK infers data dependencies between tasks from the sequential order that the tasks are inserted and the way that the data is used.
- QUARK executes the tasks using superscalar concepts in an asynchronous, dynamic fashion in order to achieve a high utilization of the available resources, using work-stealing to maintain load balance.
- QUARK relies on superscalar execution starting from serial code. The correctness of the serial code guarantees the correctness of the parallel execution.

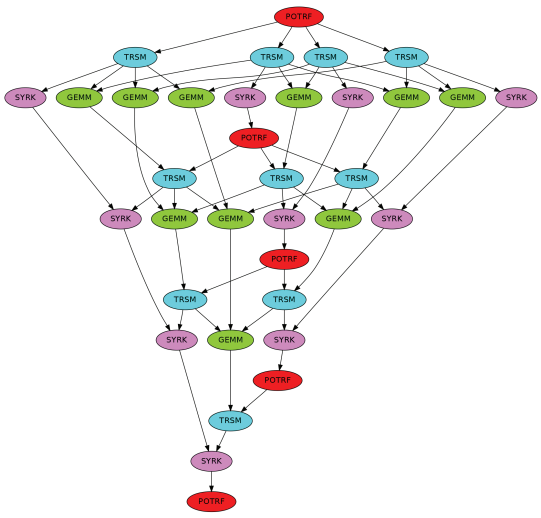
BENEFITS OF QUARK

- Simple, sequential task insertion API
- Lightweight runtime system
- Automatic dependency detection
- Dynamic task scheduling
- Scalable and tested via PLASMA
- Pipelined algorithms and DAGs
- Cooperates with other runtimes
- Can handle errors thrown by tasks
- Allows user defined task priorities
- Allows multi-threaded (gang) tasks
- Visualizes DAGs and traces

TASK-SUPERSCALAR RUNTIME

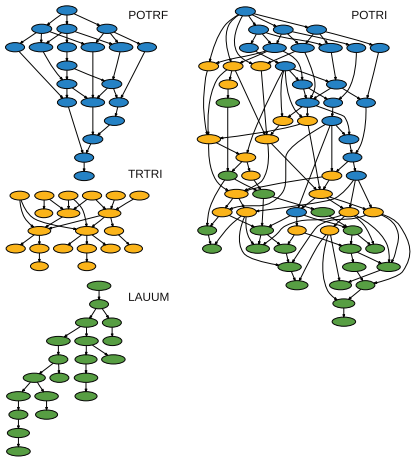
Cholesky factorization generates a sequence of tasks. In each task, data items are marked for read or write. QUARK infers the implicit data dependencies between tasks based on the sequence of tasks and the way the data is accessed. QUARK executes tasks using superscalar methods, assigning work to hardware based on data locality and load-balancing using work stealing.

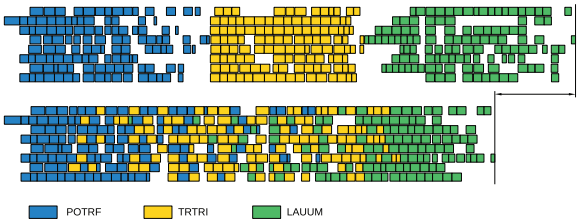




PIPELINING ALGORITHMS

Cholesky Inversion consists of calls to three algorithms (POTRF, TRTRI, LAUUM), each of which creates a DAG (shown on left). QUARK will take the DAGs and pipeline each task after the data dependencies are satisfied. This can result in an overall DAG for Cholesky Inversion (on right) that is much shorter with a shorter trace and execution time.





ICL UT INNOVATIVE COMPUTING LABORATORY THE UNIVERSITY of TENNESSEE

SPONSORED BY U.S. DEPARTMENT OF ENERGY, National Science Foundation

WITH SUPPORT FROM MathWorks, Microsoft