Recursive Serial and Parallel QR Factorization

Eric Hans Lee
Cray Math and Scientific Libraries
Cornell University Department of Computer Science

> ABSTRACT

The QR Factorization is a computational routine widely used to solve everything from least square to eigenvalue problems. It is provided in the ubiquitous Linear Algebra Package (LAPACK) via an iterative algorithm. Prior work indicates recursive algorithms lead to better performance. We present both serial and parallel implementations of the **R**ecursive **GE**neral **QR F**actorization (**rgeqrf**).

> METHODOLOGY

Our code is written in C with calls to LAPACK Fortran Routines. The standard QR Factorization (**dgeqrf**) divides a matrix into blocks. The left-most block is factored, and then the rest of blocks are updated with a matrix multiply (**gemm**). This is repeated by shifting the leftmost block right and continuing the same process.

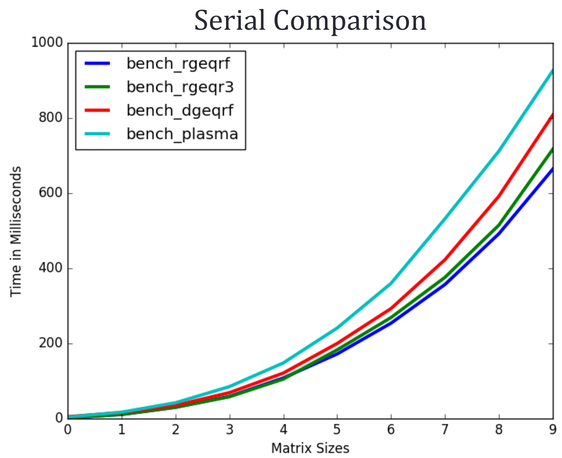
On the other hand, the recursive **rgeqrf** performs recursive block merges instead of iterative block updates. Recursive block merges recursively halve a block and combine halves when both are factored. Practically speaking, the recursive method uses more BLAS-3 operations than the iterative one, leading to quicker runtimes.

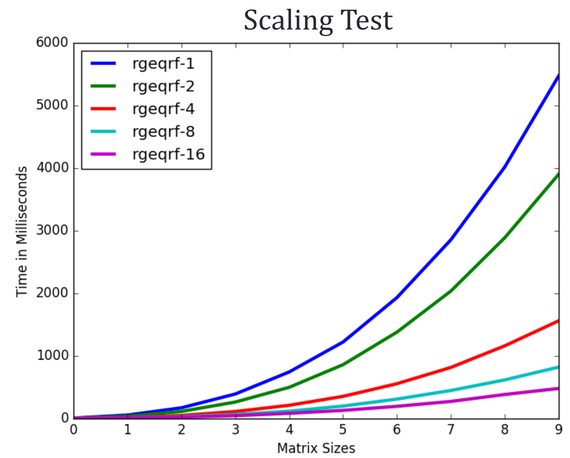
Parallelization is accomplished through synchronous openmp tasking. A matrix is divided into an **rgeqrf** (of size **ib**) portion to factor and **gemm** portions (of size **nb**) to update, similar to the original **dgeqrf**. Load balancing threads can be done either with heuristics or flop balancing

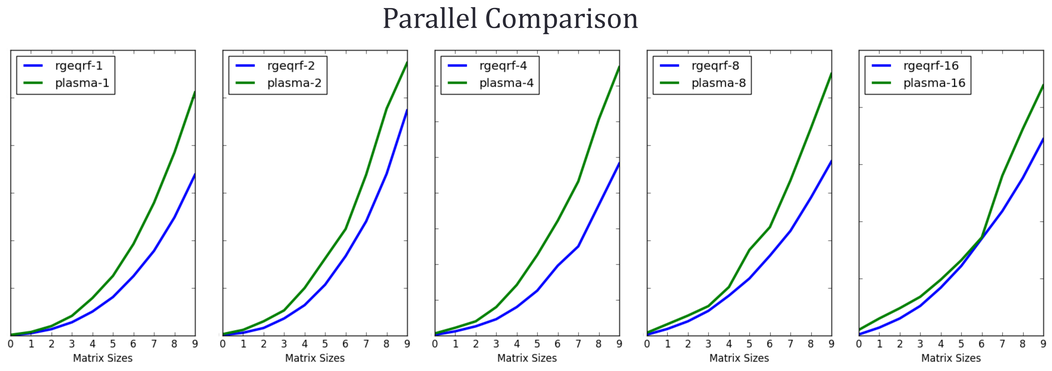
> CONCLUSIONS AND FUTURE WORK

On tall skinny matrices, **rgeqrf** performs up to 30% faster than **dgeqrf.** On square matrices, **rgeqrf** has roughly comparable performance to **dgeqrf**. We also benchmarked **plasma_dgeqrf**, another of algorithm using block-tiling, which **rgeqrf** beats as well. We perform a **rgeqrf** scaling test –scaling is quite good; this is due to the fact that parallelization of **rgeqrf** is straightforward. To improve scaling, future work includes writing an asynchronous version to remove sync time between threads.

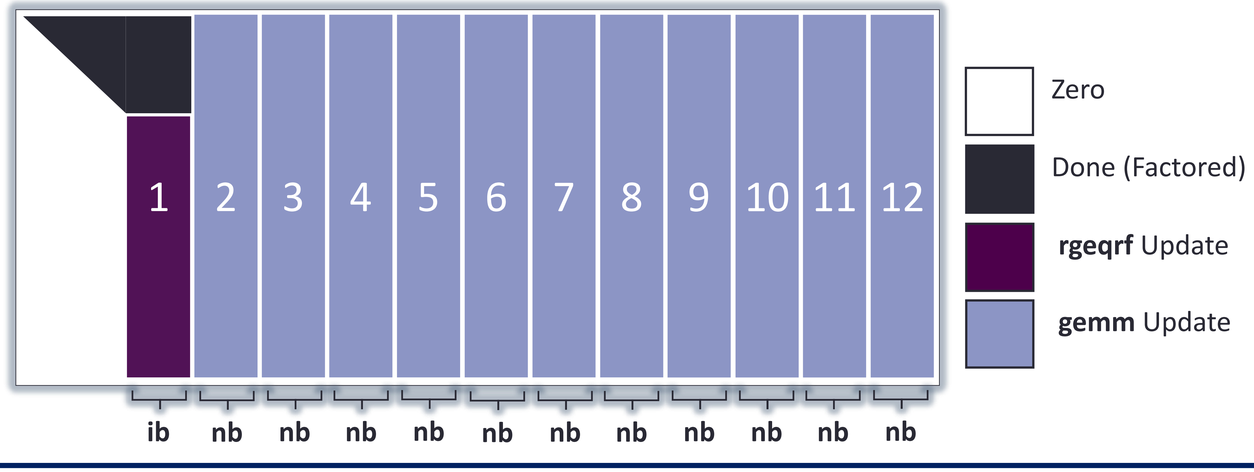
> BENCHMARKS

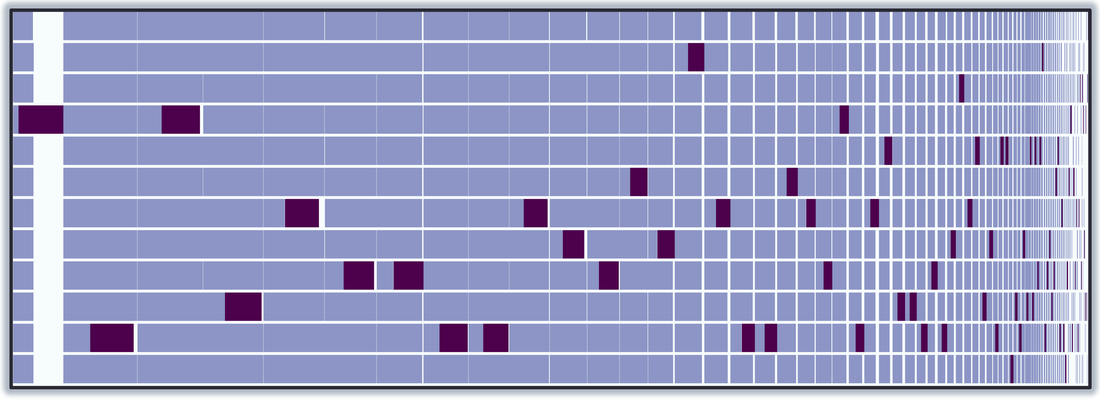






> WORK DIVISION AMONG THREADS





Parallel **rgeqrf** Execution Trace