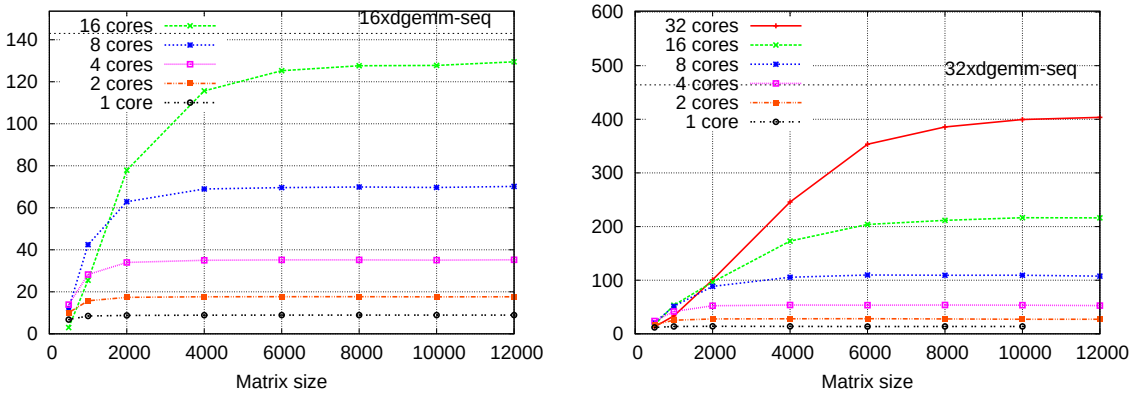
III.1. The multicore computing systems - promises and challenges: synopsis

Multicore and heterogeneous computing technology has visibly penetrated the global market. Computer manufacturers have already embarked on the multicore roadmap, promising to double the number of processors on a chip every other year, and manycores are on the horizon. This dramatic transformation of the computing landscape also include the emergence of more powerful processing elements such as GPUs, FPGA, etc [3, 4]. The shift to an increasing number of cores and heterogeneous architectures requires significant modification to today's computational tools and technologies. For at least two decades, we have taken it for granted that each successive generation of microprocessors would, either immediately or after minor adjustments, make our old software run substantially faster. But this "free ride" is about to an end. The communication costs of an algorithm can already exceed arithmetic costs by orders of magnitude, and the gap is growing exponentially over time [4]. It is difficult to overestimate the magnitude of the discontinuity that the high-performance computational physics community is about to experience because of the emergence of the next generation of multicore and heterogeneous processor designs.

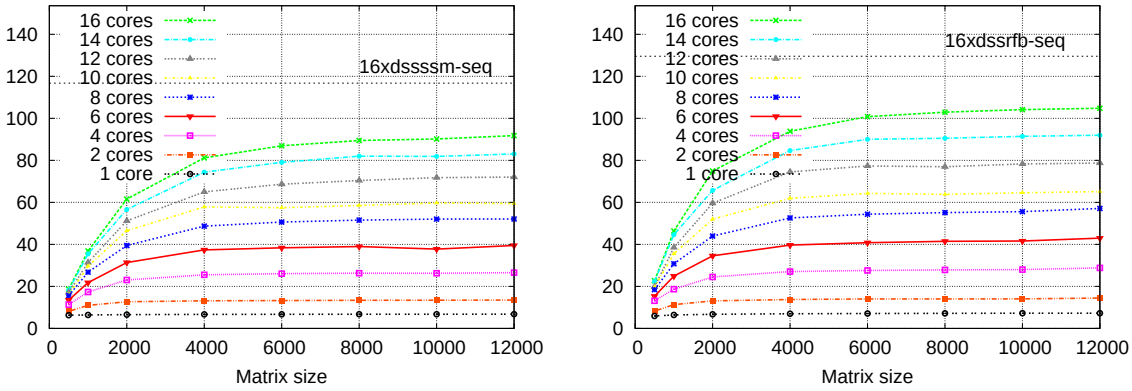
At present, the widely available linear algebra package LAPACK relies on parallel implementations of basic linear algebra subroutines (BLAS) to take advantage of multiple execution units. This approach is characterized by a fork-join model of parallel execution. On current and future generations of multicore processors, this approach may result in suboptimal performance since it introduces strict dependencies due to the presence of non-parallelizable portions of code. There are a number of efforts aiming to address the resulting critical and highly disruptive situation that is facing the linear algebra and high performance computing community. The Parallel Linear Algebra for Scalable Multi-core Architecture (PLASMA) project is one such effort [1]. The following figures show the great performance of the matrix multiplication of Intel MKL library on an Intel Xeon EMT64 (left), and of IBM ESSL library on an IBM Power 6 (right):



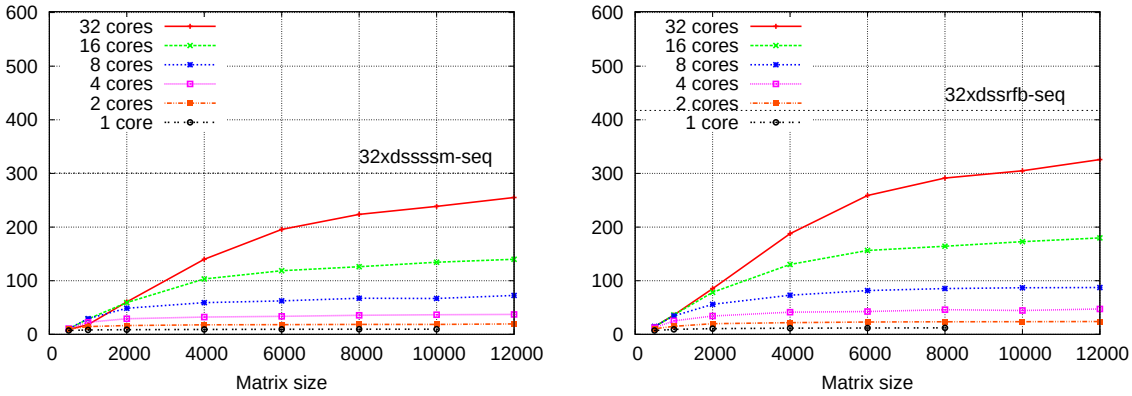
courtesy of Innovative Computing Laboratory at the University of Tennessee, Knoxville.

The following figures shows the performance of the LU factorization and QR decomposition of PLASMA.

PLASMA_DGETRF and DGEQRF on Intel Xeon EMT64

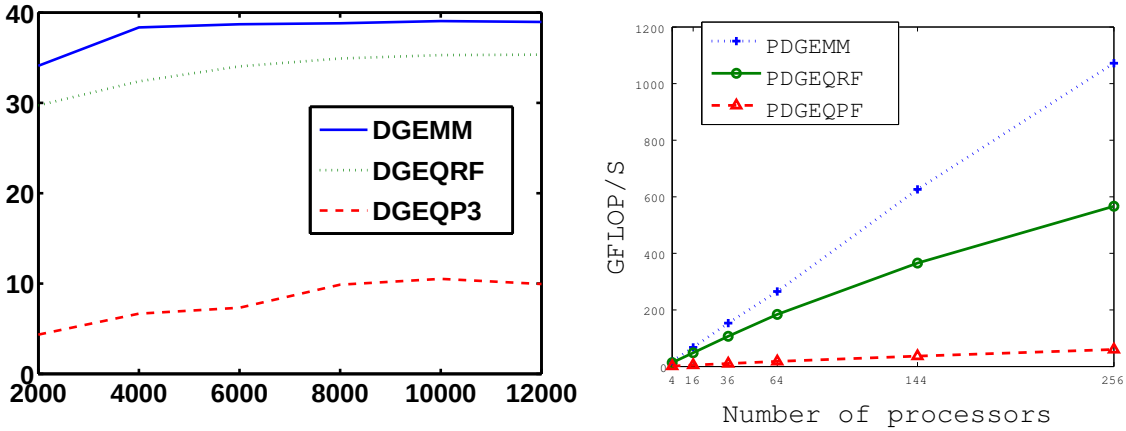


PLASMA_DGETRF and DGEQRF on IBM POWER 6



courtesy of Innovative Computing Laboratory at the University of Tennessee, Knoxville.

Unfortunately, by nature, not every matrix operation can be efficiently parallelized and match the efficiency. For example, the QR decomposition (QRD) with column pivoting is significantly poorer than the QRD without pivoting, as shown in the following plots



The performance of DGEMM (matrix multiplication), DGEQRF (QRD), and DGEQP3 (pivoted QRD) on Intel Core i7 Quad 2.66G with MKL 10.2 (left), and on Cray XT4 at NERSC (right).

This is due to the fact that the pivoting involves moving the columns of the matrix between levels of a memory hierarchy and/or between processors over a network. The communication costs can already exceed arithmetic costs by orders of magnitude, and the gap is growing exponentially over time [4].

A challenge to numerical linear algebra community is how to harvest the great performance of basic matrix operation kernels such as matrix multiplication and QR decomposition (without pivoting) and re-design higher-level matrix algorithms to use only highly parallelized linear algebra building blocks in realistic large scale physical and engineering simulation applications [2].

References

[1] E. Agullo, J. Demmel, J. Dongarra, et al, "Numerical Linear Algebra on emerging architectures: The PLASMA and MAGMA projects", Journal of Physics: Conference Series, Vol. 189, 2009

[2] J. Demmel, Avoiding communication in linear algebra, 2009. Available at http://www.cs.ucdavis.edu/∼demmel

[3] H. Sutter. A fundamental turn toward concurrency in software, Dr. Dobb's Journal, 30(3), 2005

[4] S. L. Graham, M. Snior and C. A. Patterson. Getting up to speed: the future of supercomputing. National Academies Press, Washington DC, 2005 (354 pages)