BLAS

- Basic operations with vectors and matrices dominates scientific computing programs
- To achieve high efficiency and clean computer programs an effort has been made in the last few decades to standardize and optimize such operations
- Basic Linear Algebra Subprograms
- http://www.netlib.org/blas/
- The BLAS (Basic Linear Algebra Subprograms) are routines that provide **standard** building blocks for **efficiently** performing basic vector and matrix operations. (1979)

BLAS

- Because the BLAS are efficient, portable, and widely available, they are commonly used in the development of high quality linear algebra software, LAPACK for example.
- BLAS serve as building blocks in many computer codes, and their performance on a certain computer usually reflects the performance of that code on the same computer.
- This is why most of the computer vendors optimize BLAS for specific architectures.

BLAS

- Taxonomy based on computational complexity = number of needed Floating Points operations to complete an operation
- The Level 1 BLAS perform scalar, vector and vector-vector operations, the Level 2 BLAS perform matrix-vector operations, and the Level 3 BLAS perform matrix-matrix operations.
- Inner product of two vectors of size n : n multiplication and (n-1) additions for a total of approximately $2n$ operations
- Matrix-Vector multiplication: n^2 mult and $n(n-1)$ add $\approx 2n^2$ ops
- Matrix-Matrix multiplication $\approx 2n^3$

Level 3 Mat-Mat Multiplication

- Inner (dot) product version:

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix} = \begin{bmatrix} 1 \cdot 5 + 2 \cdot 7 & 1 \cdot 6 + 2 \cdot 8 \\ 3 \cdot 5 + 4 \cdot 7 & 3 \cdot 6 + 4 \cdot 8 \end{bmatrix}$$

- Middle (*saxpy*) product version:

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix} = \begin{bmatrix} 5 \begin{bmatrix} 1 \\ 3 \end{bmatrix} + 7 \begin{bmatrix} 2 \\ 4 \end{bmatrix} & 6 \begin{bmatrix} 1 \\ 3 \end{bmatrix} + 8 \begin{bmatrix} 2 \\ 4 \end{bmatrix} \end{bmatrix}$$

- Outer product version:

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix} = \begin{bmatrix} 1 \\ 3 \end{bmatrix} \begin{bmatrix} 5 & 6 \end{bmatrix} + \begin{bmatrix} 2 \\ 4 \end{bmatrix} \begin{bmatrix} 7 & 8 \end{bmatrix}.$$

ATLAS

- The most recent trend has been todevelop a new generation of “self-tuning” BLAS libraries targeting the rich but complex memory systems of modern processors (1996)
- Automatically Tuned Linear Algebra Software
- http://math-atlas.sourceforge.net/
- ATLAS is an implementation of empirical optimization procedures that allow for many different ways of performing a kernel operation with corresponding timers to determine which approach is best for a particular platform.
- Multiple implementation : different hand-written versions of the same kernel are explicitly provided

ATLAS

- code generation : a highly parameterized code is written that generates many different kernel implementations (different block sizes, unit or larger stride, ...)

BLAS and Memory Access

- The practical difference in the various ways of implementing matrix-vector and matrix-matrix operations lies in the way we access memory and the type of memory, i.e., main memory or cache memory
- The cost associated with the BLAS programs can be computed by taking into account the total number of floating operations but also including the cost associated with memory access to load the operands.
- $$T = n_f \times \delta t + n_m \times \tau = n_f \times \delta t(1 + \frac{n_m}{n_f} \times \frac{\tau}{\delta t}),$$

Level 1 - "BLAS1"

$n_f = 2n$, $n_m = 3n+1$, n_m/n_f -> 3/2 if n >>

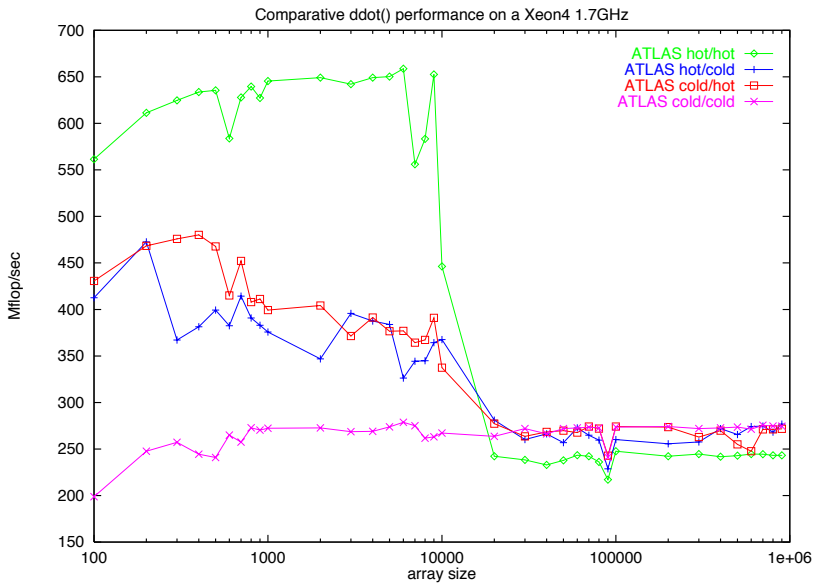

Figure 2.8: Performance of the dot product on the Intel Pentium-4 with speed 1.7 GHz. (Courtesy of C. Evangelinos)

Level 2 - "BLAS2"

$n_f = 2n^2$, $n_m = n^2+3n$, n_m/n_f -> 1/2 if n >>

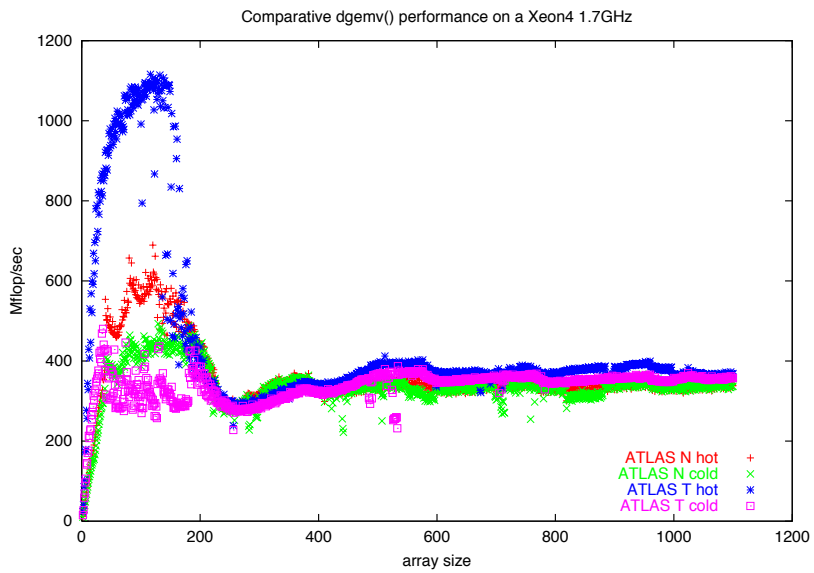

Figure 2.9: Performance of the matrix-vector multiply on the Intel Pentium-4 with speed of 1.7 GHz. (Courtesy of C. Evangelinos)

Level 3 - "BLAS3"

$n_f = 2n^3$, $n_m = 3n^2$, n_m/n_f -> 3/2n if n >>

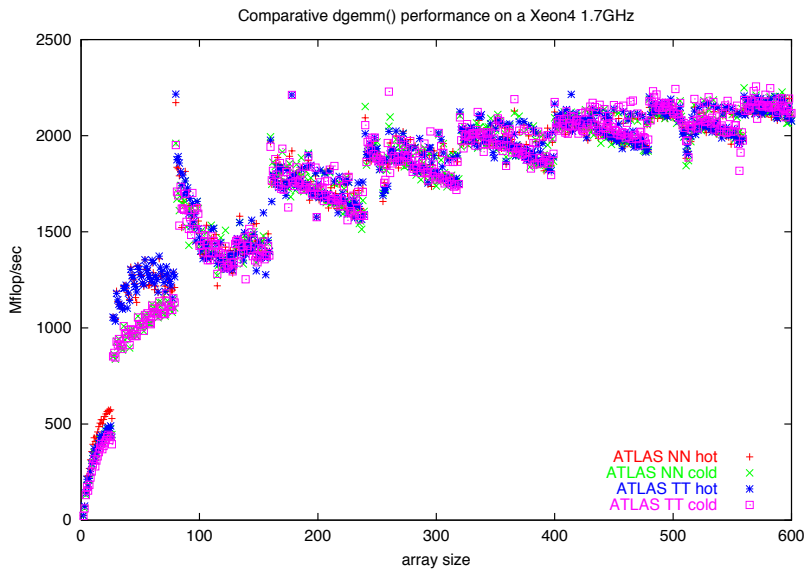

Figure 2.10: Performance of the matrix-matrix multiply on the Intel Pentium-4 with speed 1.7 GHz. (Courtesy of C. Evangelinos)

Optimized BLAS libraries

- AMD Core Math Library (ACML)
- http://developer.amd.com/tools/cpu/acml/pages/default.aspx
- Intel® Math Kernel Library (Intel® MKL)
- http://software.intel.com/en-us/intel-mkl
- EIGEN3
- http://eigen.tuxfamily.org

LAPACK

- LAPACK — Linear Algebra PACKage
- http://www.netlib.org/lapack/
- LAPACK is written in Fortran 90 and provides routines for solving systems of simultaneous linear equations, least-squares solutions of linear systems of equations, eigenvalue problems, and singular value problems. The associated matrix factorizations (LU, Cholesky, QR, SVD, Schur, generalized Schur) are also provided
- LAPACK routines are written so that as much as possible of the computation is performed by calls to the Basic Linear Algebra Subprograms (BLAS). LAPACK is designed at the outset to exploit the Level 3 BLAS

HPL-Linpack (Top500)

- A Portable Implementation of the High-Performance Linpack Benchmark for Distributed-Memory Computers
- http://www.netlib.org/benchmark/hpl/
- HPL is a software package that solves a (random) dense linear system in double precision (64 bits) arithmetic on distributed-memory computers.