AMD Core Math Library Overview – Cray User Group

May 2005

Chip Freitag

ACML- AMD Core Math Library

- **Suite of highly tuned math functions for high performance computing**
- **Version 1.0 introduced July 1, 2003**
- **Co-developed with The Numerical Algorithms Group LTD**
- **Just released AMD ACML 2.6**
- **BLAS** – Basic Linear Algebra Subprograms
  - Full Level 1, 2, and 3 support
  - Highly optimized DGEMM, other Level 3 BLAS.
  - OpenMP support for key routines
- **Lapack** – Linear Algebra package
  - Uses calls to BLAS to solve linear algebra systems
  - Given Matrix A, Vectors x,y, Solve A*x =y for x
  - OpenMP support for key routines
- **FFTs** – Fast Fourier Transforms
  - Time-to-frequency domain
  - Hand-tuned assembly
  - OpenMP support for 2D, 3D transforms
- **Double, Single, Complex**
- **Fast / vector transcendental math library**
  - 1, 2, 4, or N values per call
  - Single, Double precision (IEEE754)

ACML 2.6 Release

- Went live on April 29, 2005
- Dual core support
  – Linpack scales very well
- L3 BLAS tuning
  – improvements in dgemm, dgeqrf, dpotrf, dsymm
  – 91% efficiency on dgemm
- Lapack end user tuning
  – Ability to choose optimal ilaenv parameters based on application needs.
- Vector Math Library
  – Added logf, log10, powf
  – In addition to sin, cos, sincos, exp, expf, log, pow

Vector Math Library

- Reimplementation of C routines in AMD64 assembly, SSE/SSE2 instructions
  - Double precision scalar, 2-value, 4-value, array
  - Single precision scalar, 4-value, 8-value, array
  - Accuracy – better than 1 ULP (2 ulps for sin/cos)
- Estimated performance in CPU clocks*

Name	GNU libm for AMD64 (scalar)	Fast Scalar	Vector Per/value	Array (n=24)
Exp	116	75	40	26
Expf	103	81	23	15
Log	122	96	57	48
Logf	121	80-90	31	26
Log10	137	110	72	54
Pow	540	200	na	na
Powf	330	175	100	na
Cos/Sin	120	88-104	60-65	43-51
Sincos	N/A	99-114	77	45

- Caveats
  - C99 error return values
  - No system error handling
  - Denormals may cause unpredictable results
- More routines, about 1 per quarter
  - acos, asin, atan, atan2, tan, tanh, cosh, sinh

* Results obtained with AMD Opteron™ processor Rev C, 512MB DDR400 memory

ACML 3.0

- 3.0 release for ISC 2005
- Random Number Generators
- FFT radix plan builder
  - An automated mechanism to choose optimal radices based on problem size.
- Vector Math Library
  - sinf/cosf, powx

ACML 3.0 – Features

- **Random Number Generators**
  - Five base generators
    - 2 Short period
      - NAG basic, Wichmann-Hill
    - 2 Long period
      - L'Ecuyer combined multiple recursive, Matsumoto Mersenne Twister
    - 1 Cryptographically secure
      - Blum-Blum-Shub
  - 25 distributions
    - 15 univariate continuous
    - 7 univariate discrete
    - 3 multivariate
  - Other features
    - Ability to choose any of the BRNGs for the non-uniform distributions
      - Including user supplied
    - Independent streams
    - Ability to return multiple variates from a given distribution (vectors)
    - Save, copy, restore state of the generator.
  - Interface similar to SPRNG
  - Testing done with DIEHARD, NIST, and FIPS 140-2 suites.

ACML FFT Support

- Complex, Real-Complex, Complex-Real
- Simple interfaces
  - In-place transforms, fixed scaling factors
- Expert interfaces offer more control
  - scaling
  - in-place, out-of-place
  - more flexibility in describing input, output arrays.
- 1D, 2D, and 3D routines.

ACML FFT examples

- 1 dimensional real-complex, simple interface
- **DZFFT** (*MODE,N,X,COMM,INFO*)
  - Integer **MODE** – input, specifies operation
    - *MODE*=0 : only initializations (specific to the value of *N*); this is usually followed by calls to the same routine with *MODE*= 1.
    - *MODE*=1 : real transform. Initializations are assumed to have been performed by a prior call to DZFFT.
    - *MODE*=2 : initializations and a real transform.
  - Integer **N**
    - length of input real sequence X
  - Double precision **X**
    - On input: real sequence of length *N*.
    - On output: transformed Hermitian sequence.
  - DOUBLE PRECISION **COMM**(*3*N+100*)
    - communication or work array
  - INTEGER **INFO**
    - *INFO* is an error indicator. On successful exit, *INFO* contains 0. If *INFO* = -i on exit, the i-th argument had an illegal value.

ACML FFT examples

- 2 dimensional complex-complex, simple interface
- **ZFFT2D** (*MODE,M,N,X,COMM,INFO*)
  - Integer **MODE** – input, specifies direction of transform
    - *MODE*=-1 : forward 2D transform.
    - *MODE*=1 : backward (reverse) 2D transform.
  - Integer **M**
    - Number of rows in the 2D array of data to be transformed. If *X* is declared as a 2D array then *M* is the first dimension of *X*.
  - Integer **N**
    - Number of columns in the 2D array of data to be transformed. If *X* is declared as a 2D array then *N* is the second dimension of *X*.
  - COMPLEX*16 **X**(*M*N*)
    - On input: *X* contains the *M* by *N* complex 2D array to be transformed.
    - On output: *X* contains the transformed sequence.
  - COMPLEX*16 **COMM**(*M*N+3*(M+N)*)
    - communication or work array
  - INTEGER **INFO**
    - *INFO* is an error indicator. On successful exit, *INFO* contains 0. If *INFO* = -i on exit, the i-th argument had an illegal value.

ACML FFT examples

- 2 dimensional complex-complex, expert interface
- **ZFFT2DX** (*MODE,SCALE,LTRANS,INPL,M,N,X,INCX1, INCX2,Y,INCY1,INCY2,COMM,INFO*)
  - Integer **MODE** – input, specifies operation of transform
    - *MODE*=0 : only initializations
    - *MODE*=-1 : forward transform. Initializations are assumed to have been performed by a prior call to ZFFT2DX.
    - *MODE*=1 : backward (reverse) transform. Initializations are assumed to have been performed by a prior call to ZFFT2DX.
    - *MODE*=-2 : initializations and a forward transform
    - *MODE*=2 : initializations and a backward transform
  - DOUBLE PRECISION **SCALE**
    - Scaling factor to apply to the output sequences
  - LOGICAL **LTRANS**
    - if *LTRANS* is .TRUE. then a normal final transposition is performed internally to return transformed data consistent with the values for arguments *INPL*, *INCX1*, *INCX2*, *INCY1* and *INCY2*
    - If *LTRANS* is .FALSE. then the final transposition is not performed
  - LOGICAL **INPL**
    - if *INPL* is .TRUE. then *X* is overwritten by the output sequences; otherwise the output sequences are returned in *Y*.

Linux 64-bit OS

SLES 8, SLES 9, SL9.x, RHEL 3

Compiler	***Version (ACML build)***	***OpenMP support***
GNU	3.3	no
PGI	6.0-3	yes
Pathscale	2.1	planned

Linux 32-bit OS
SLES 8, SLES 9, SL9.x

Compiler	***Version (ACML build)***	***OpenMP support***
GNU noSSE noSSE2	3.3	no
PGI noSSE noSSE2	6.0-3	yes
Pathscale noSSE noSSE2	2.1	planned

ACML C interfaces

- C prototypes for calls to the Fortran routines

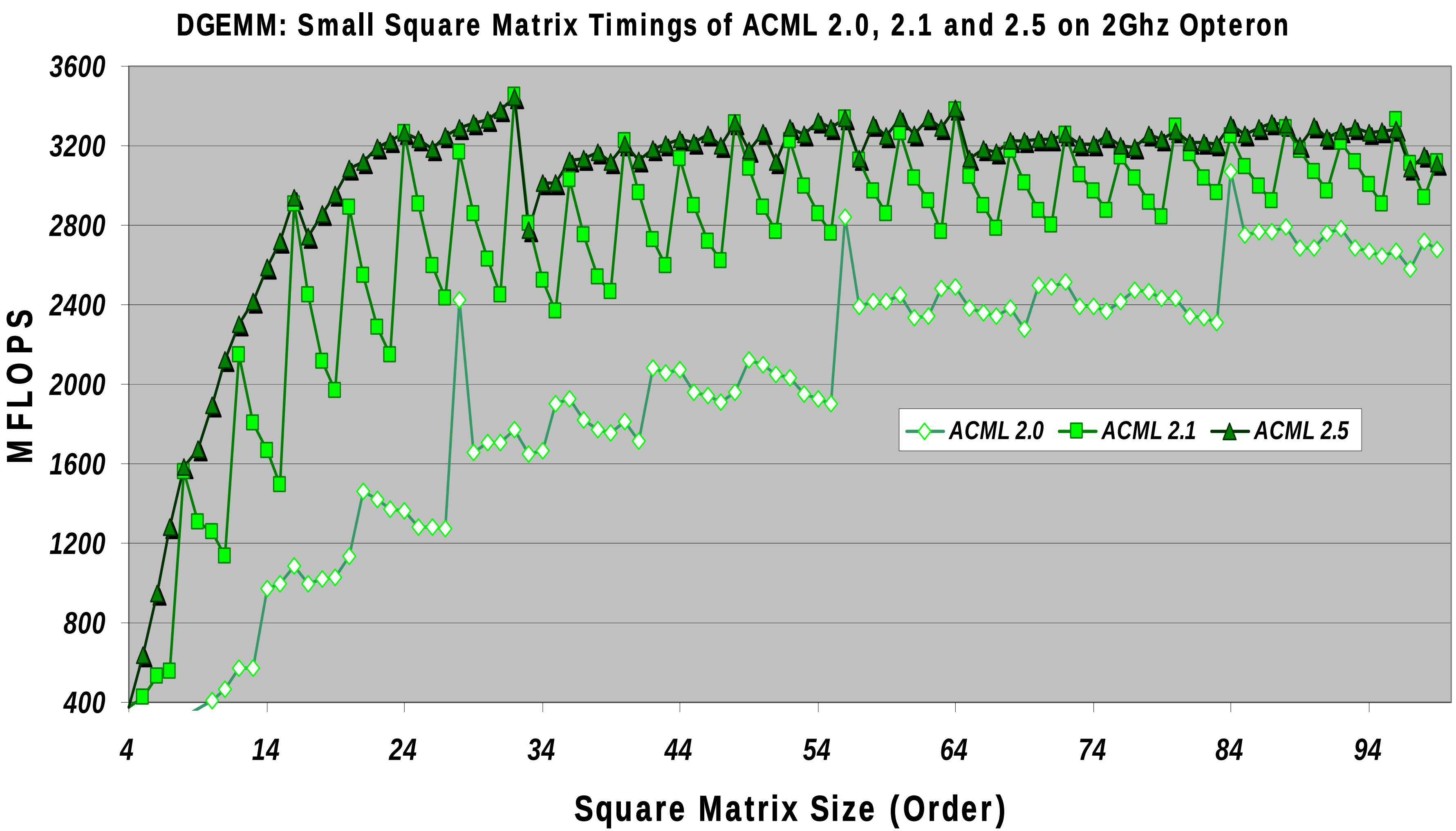
- C style interfaces provided for all routines
  - No workspace arguments – all workspace allocated by C interface routines
  - Column-major ordering
  - acml.h
    - Complex type definitions
    - complex data initialization, basic math
    - BLAS
    - LAPACK
    - FFTs
    - Sparse BLAS
  - Fortran names have trailing _, Upper case name

ACML Sparse BLAS

- Level 1 routines
- Sparse vector x in compressed form, full form y.

64-ACML 2.5 Snapshot

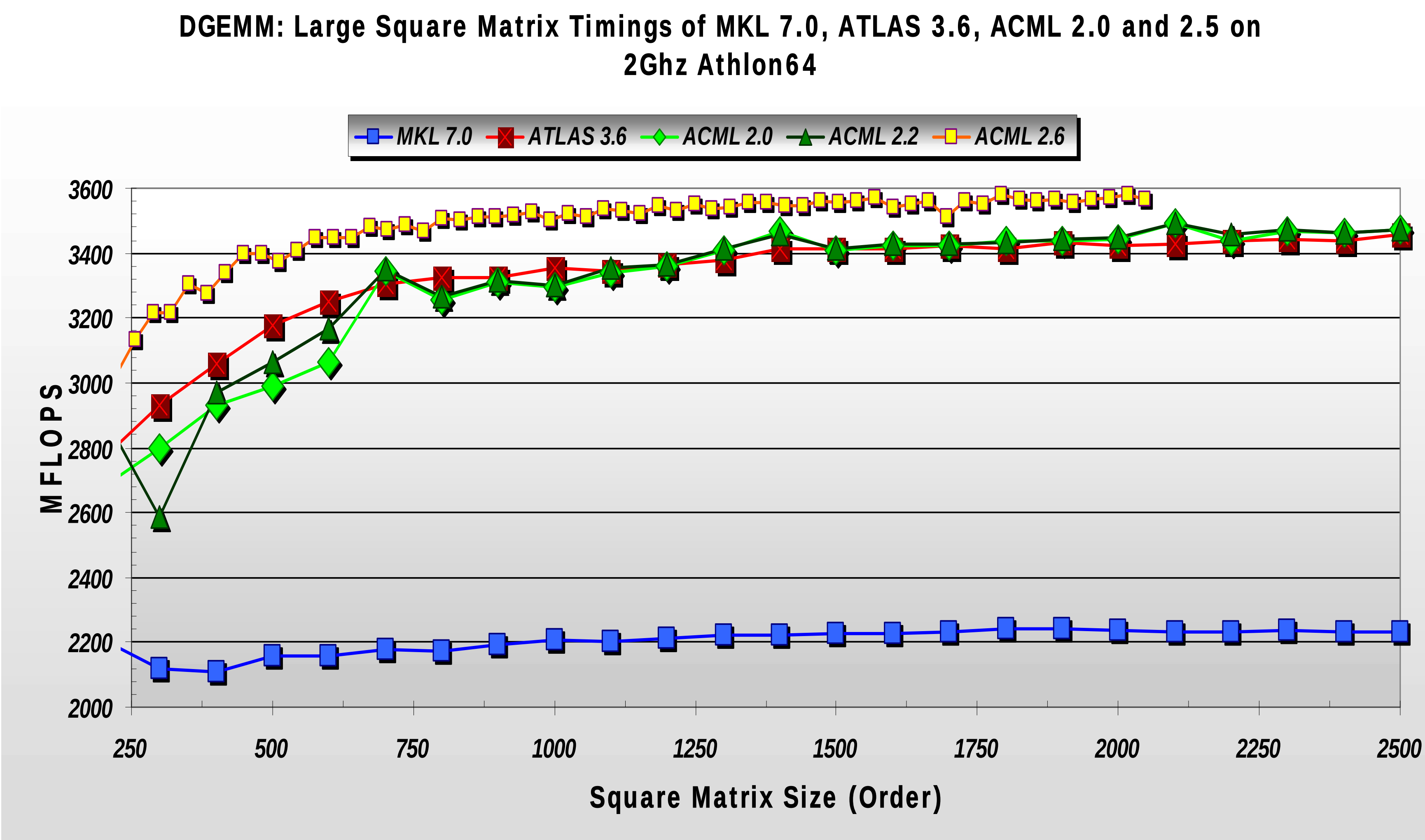
Small Dgemm Enhancements



* Results obtained with AMD Opteron™ processor Rev C, 8GB DDR400 memory

64-ACML 2.5 Snapshot

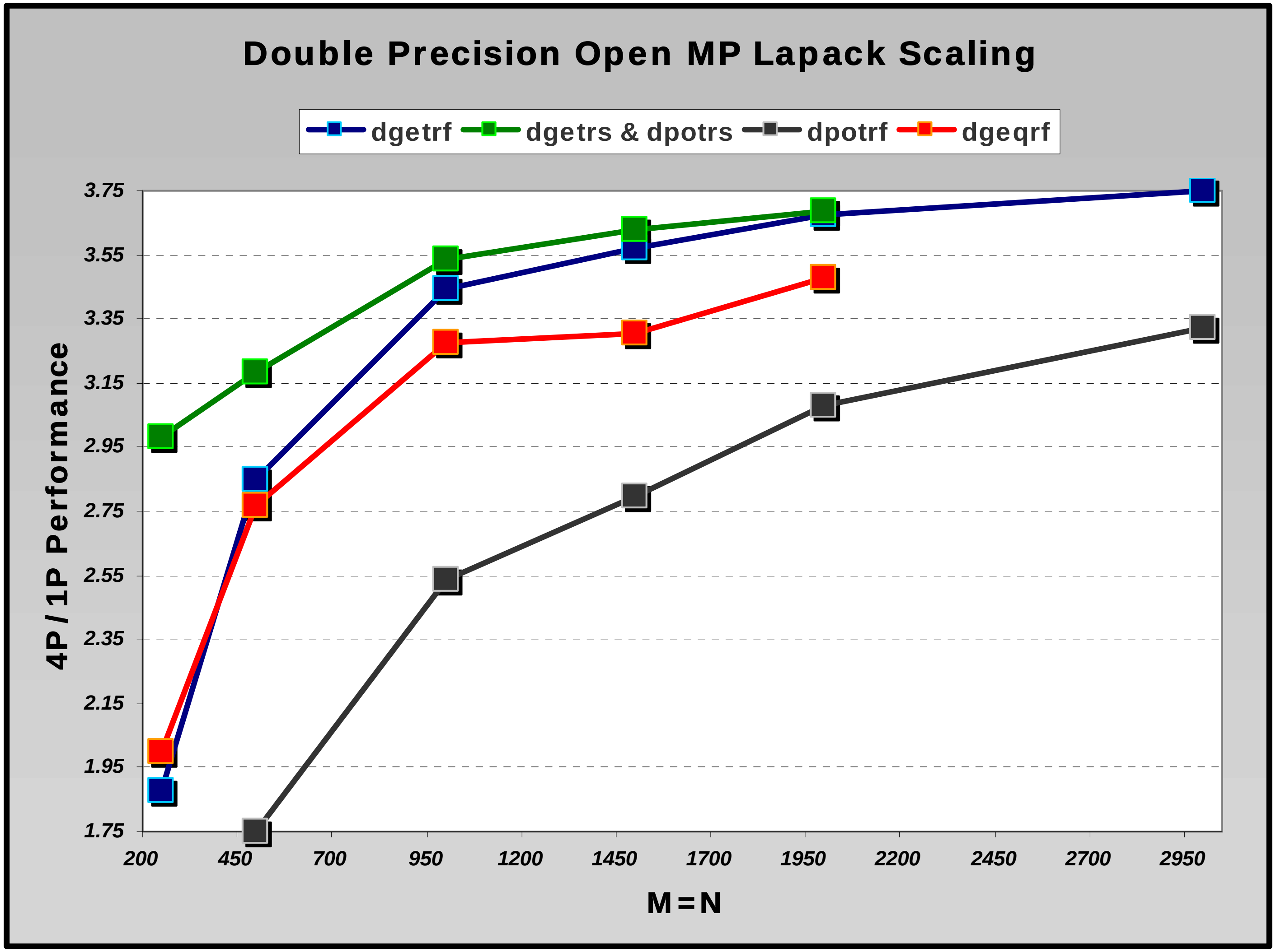
Large Dgemm Enhancements



* Results obtained with AMD Opteron™ processor Rev C, 8GB DDR400 memory

Multithreaded LAPACK Performance

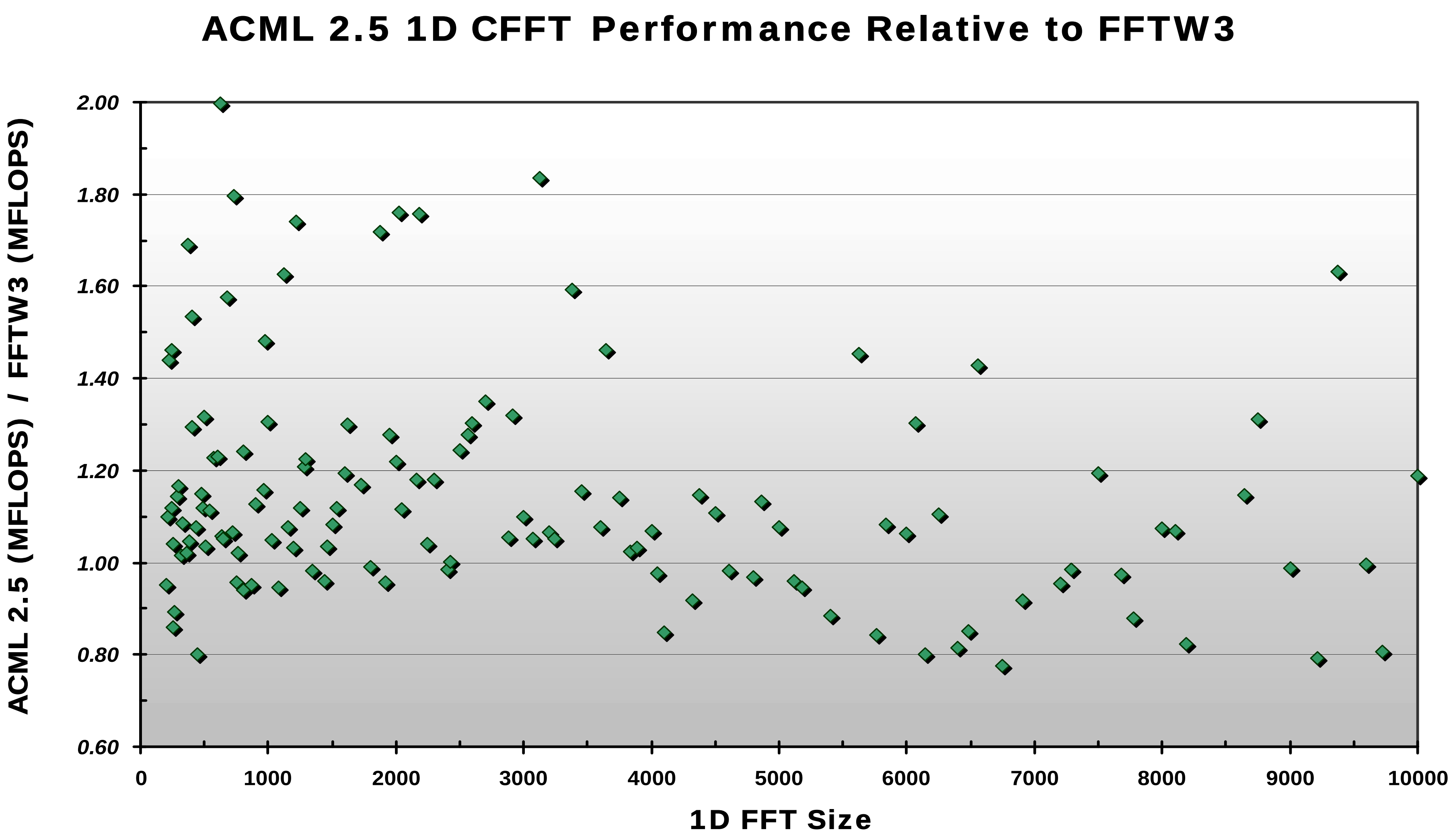
Double Precision (LU, Cholesky, QR Factorize / Solve)



* Results obtained with ACML2.0 on 4 x AMD Opteron™ processor Rev C, 4GB DDR333 memory

1D FFT Performance (non-power of 2)

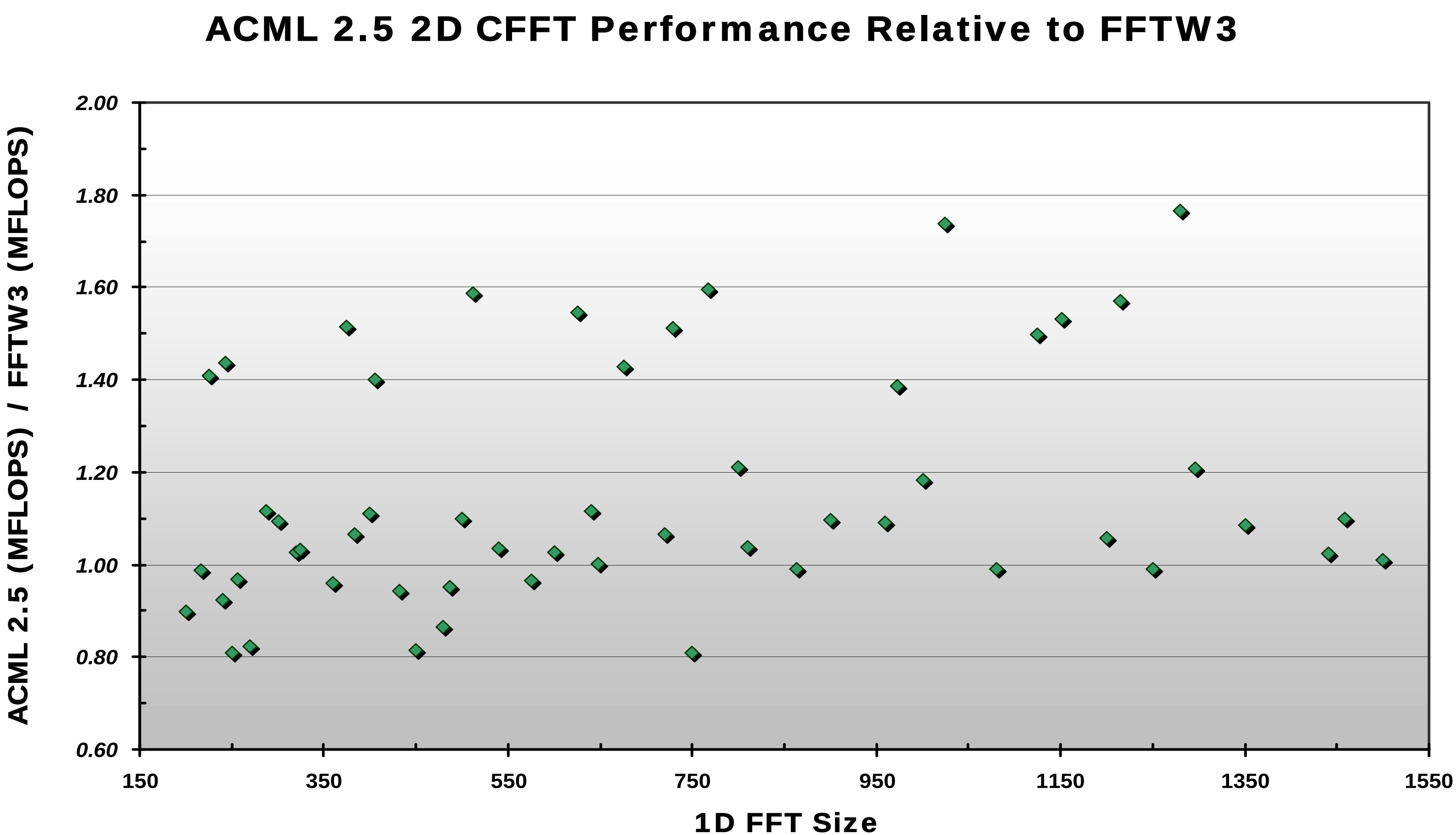
ACML 2.5 vs FFTW 3.1



* Results obtained with AMD Opteron™ processor Rev C, 512MB DDR333 memory

2D FFT Performance (non-power of 2)

ACML 2.5 vs FFTW 3.1



* Results obtained with AMD Opteron™ processor Rev C, 512MB DDR333 memory