



The Numerical Algorithms Group
Combining mathematics and technology for
enhanced performance

Design and Performance of the AMD ACML

Dr Edward Smyth

Over 30 years of numerical excellence

Brief Introduction to NAG

- Founded 1970
 - Co-operative software project
- Incorporated 1976
 - NAG Ltd (UK)
 - not-for-profit organisation
- ~\$10 million financial turnover
- ~70 employees
- Main product still the NAG Libraries
 - Also Visualisation, Data Mining Components, Compiler, ...

Over 30 years of numerical excellence

NAG Numerical Libraries

- NAG provides high-level math and stats components
 - Nonlinear equation solvers
 - Summation of series and transformations, FFTs
 - Quadrature
 - ODEs, PDEs and integral equations
 - Approximation and curve and surface fitting
 - Optimization and operations research
 - Dense linear algebra, including LAPACK
 - Sparse linear systems and eigenproblems
 - Special functions
 - ...
- Building upon Vendor-supplied basic math libraries

Over 30 years of numerical excellence

June 2004

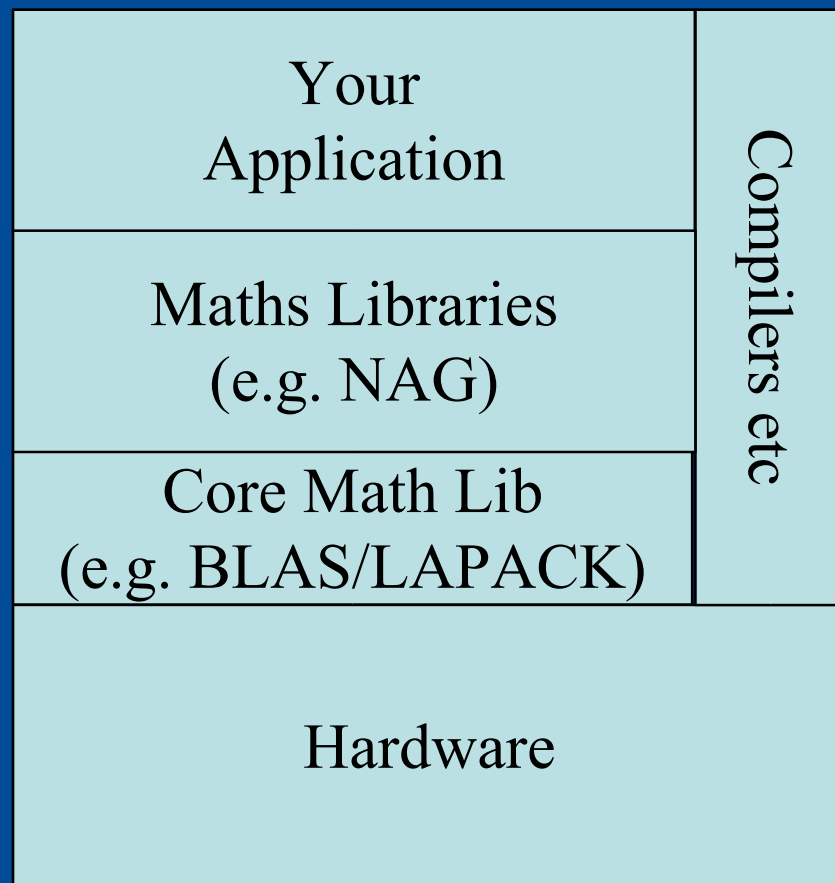
Sun HPC Consortium, Heidelberg

3

NAG[®]

The Importance of Math Libraries in Development – Performance (& Accuracy)

- Multiplicative effect of improvements!
 - Great hardware advances...
 - Better compilers
 - Optimise the core
 - Provide/optimize useful functionality
 - Take full advantage of any future improvements at ANY level!



AMD Core Math Library (ACML)

- NAG collaborating with AMD to produce ACML — the AMD Core Math Library.
 - 1.0 announced at LinuxWorld in January 2003
 - 1.5 at SC'03 in November 2003
 - 2.0 at Cluster World in April 2004
- Contents:
 - BLAS – Level 1, 2 and 3
 - LAPACK – all routines and drivers from LAPACK 3
 - FFTs – a selection of routines
- Tuned to run well on Opteron™ processors
- Extensive test/timing harness

Over 30 years of numerical excellence

Implementation of ACML

- 32- and 64-bit versions
- Fortran and C interfaces
- Linux and Windows
- Safe for use by multi-threaded programs
- Increasingly adding multi-threaded versions
- Hand-coded assembly language kernels

ACML Tuning

- Use vectorized SSE instructions

e.g. `mulps %xmm1, %xmm2`

`xmm1` and `xmm2` are 128-bit registers, each holding four 32-bit floating-point numbers. The `mulps` instruction multiplies them elementwise:

<code>xmm1</code>	a1	a2	a3	a4
<code>xmm2</code>	b1	b2	b3	b4
<code>xmm1*xmm2</code>	a1*b1	a2*b2	a3*b3	a4*b4

- Arrange data so it can be operated on as above – cleanup loops are likely to be required

ACML Tuning (continued)

- Use prefetching where appropriate
 - Ensure that data is loaded into cache memory before it is used
 - For performance, arrange algorithm so that everything in cache is used – not just part of it
 - AMD64 chips have hardware prefetching, which can be hurt by explicit prefetching – need to take care
- Unroll loops where possible
- Arrange data loads and stores to minimize processor stalling
- Experiment to choose optimal data storage

ACML Tuning (continued)

- Key algorithms used by LAPACK routines completely redesigned to take maximum advantage of multi-processor SMP systems
 - New blocking algorithms designed from scratch
 - Easily outperforms standard (netlib) LAPACK
 - No compromise of performance on single processor
- Automatic procedure to choose best block sizes
- AMD64 Software Optimization Guide available:
<http://www.amd.com/us-en/Processors/TechnicalResources/>

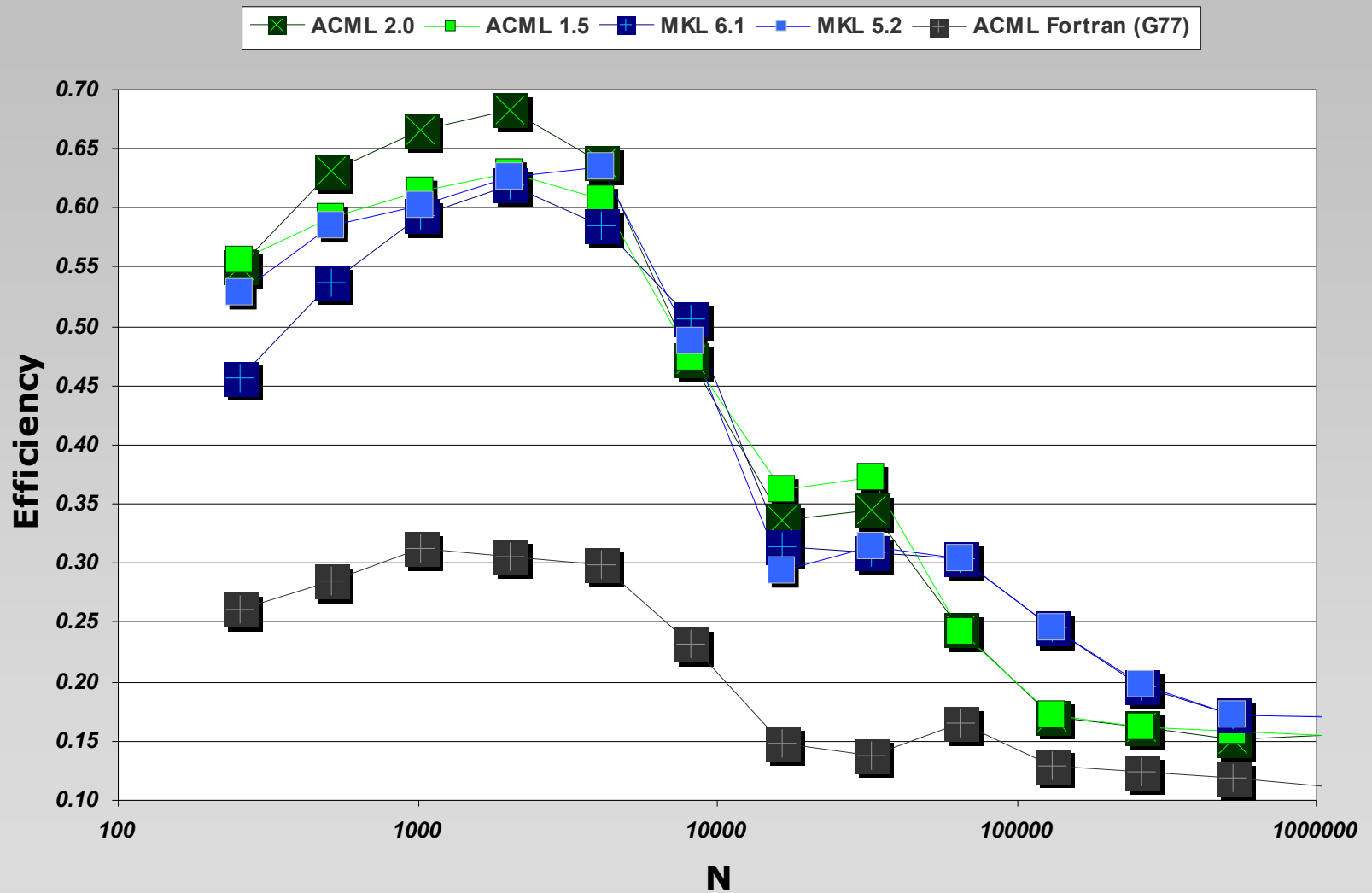
Some Timing Results

- Results by Dr Tim Wilkens, AMD – presented at Cluster World 2004 (San Jose)
- Comparing ACML 2.0/1.5 and Intel® MKL 5.2/6.1 (32-bit)
- AMD64 results generated by AMD and NAG
 - 1P Athlon64 3200+ / 512MB / 1MB L2-cache
 - 4P Opteron 844 / 4GB / 1MB L2-cache per processor
- Pentium 4 results generated by AMD
 - 1P Pentium4 1.7GHz

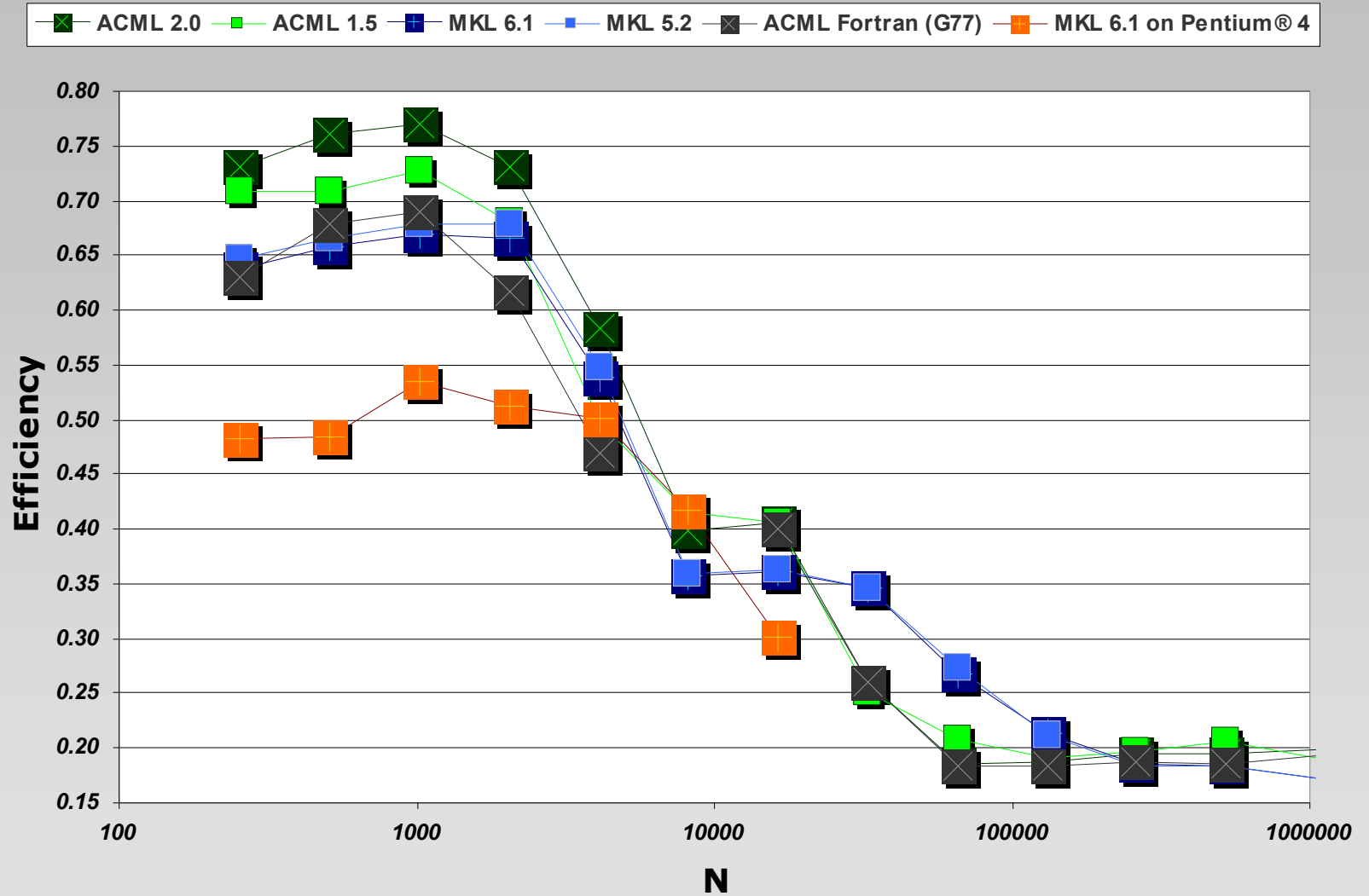
FFTs

- Initial releases were Fortran-only, but slow compared to FFTW
- Latest releases include assembly kernels tuned for multiple radices
 - For ACML 2.0, non-powers-of-2 saw a significant improvement

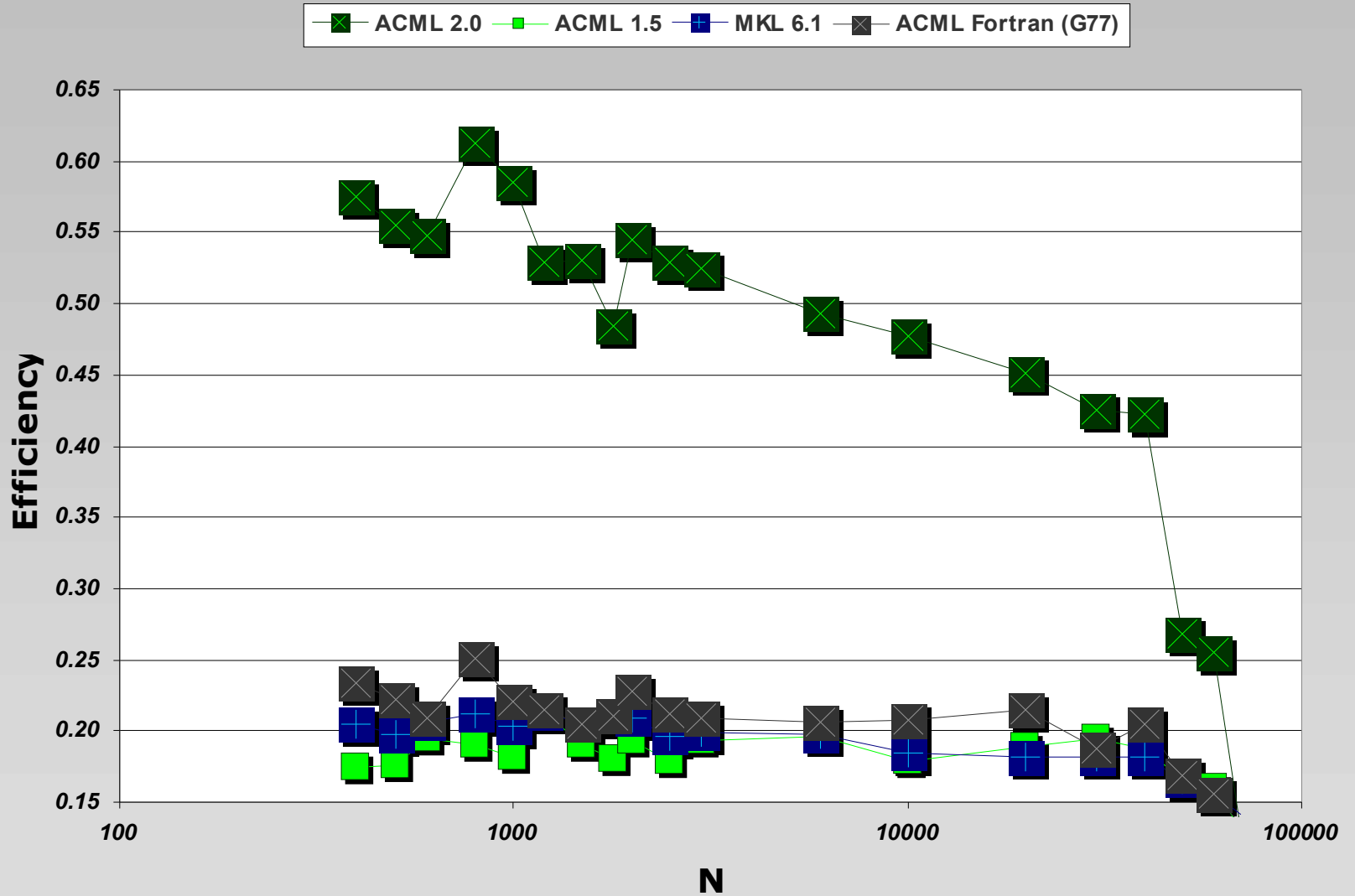
CFFFT 1D (powers of 2)



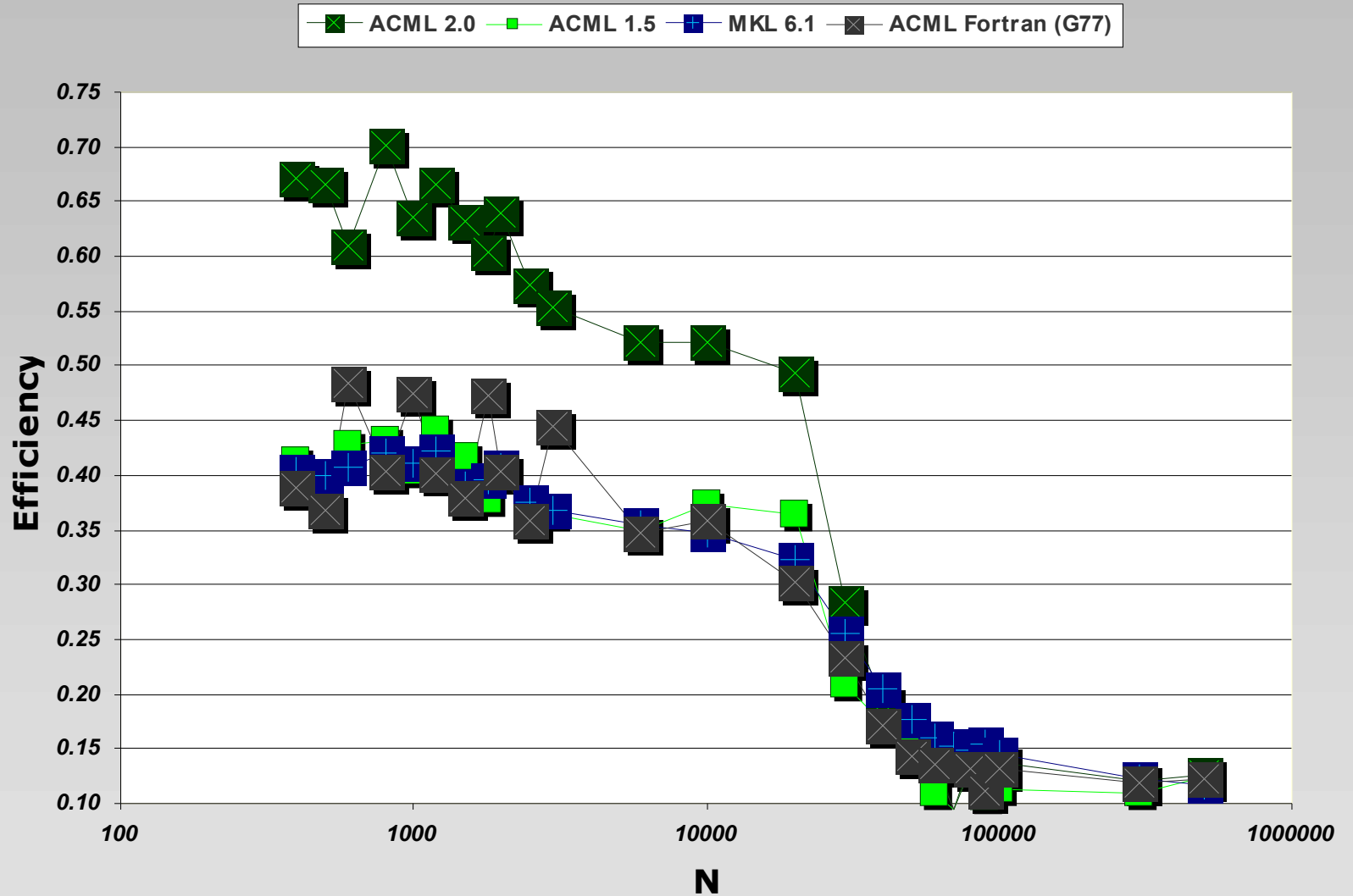
ZFFT 1D (powers of 2)



CFFT 1D (non-powers of 2)



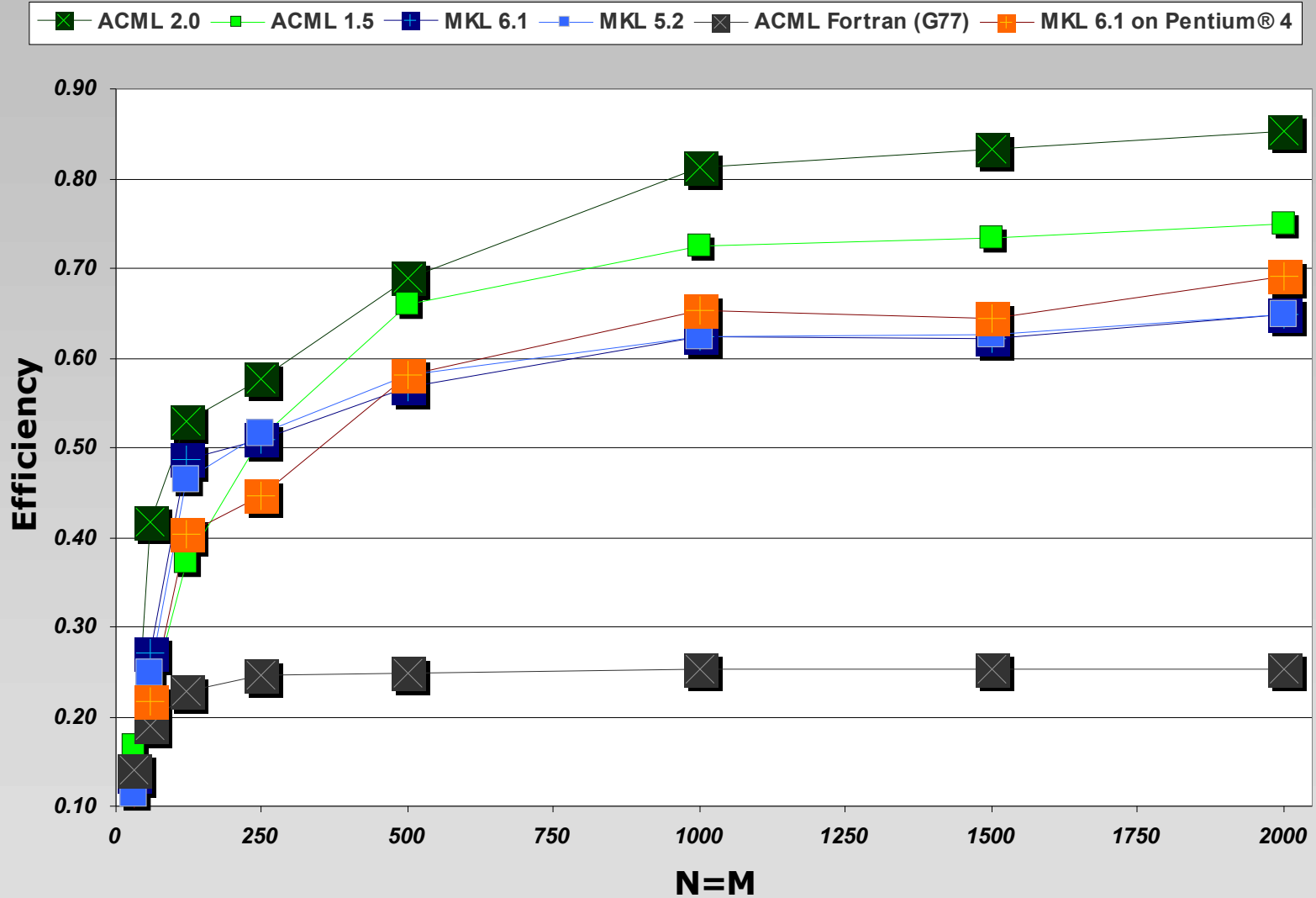
ZFFT 1D (non-powers of 2)



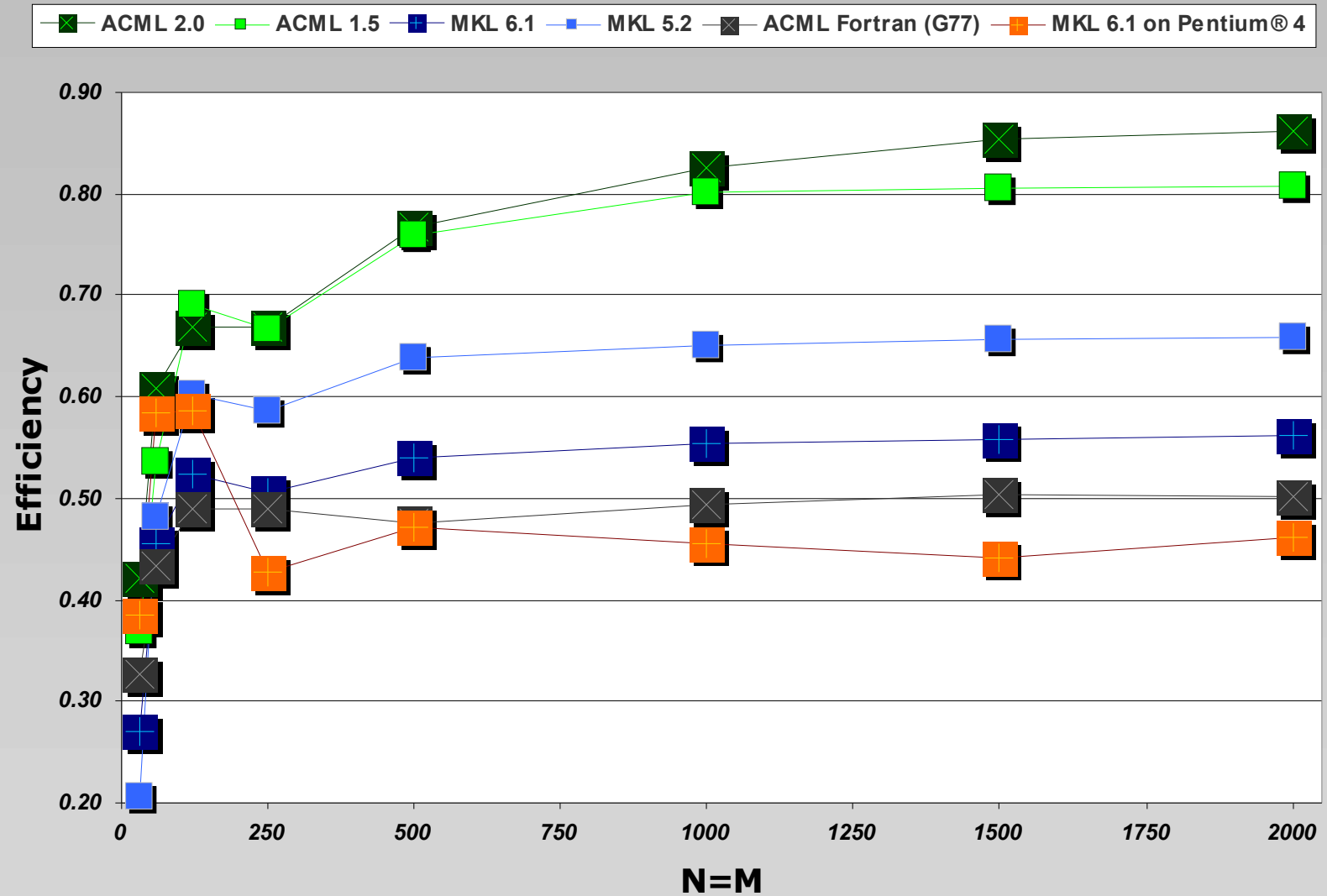
BLAS

- Key BLAS need to be aggressively tuned with assembly (cf Fortran-only!)
 - To benefit higher level routines
 - To compete!

SGEMM (SP general matrix matrix multiply)



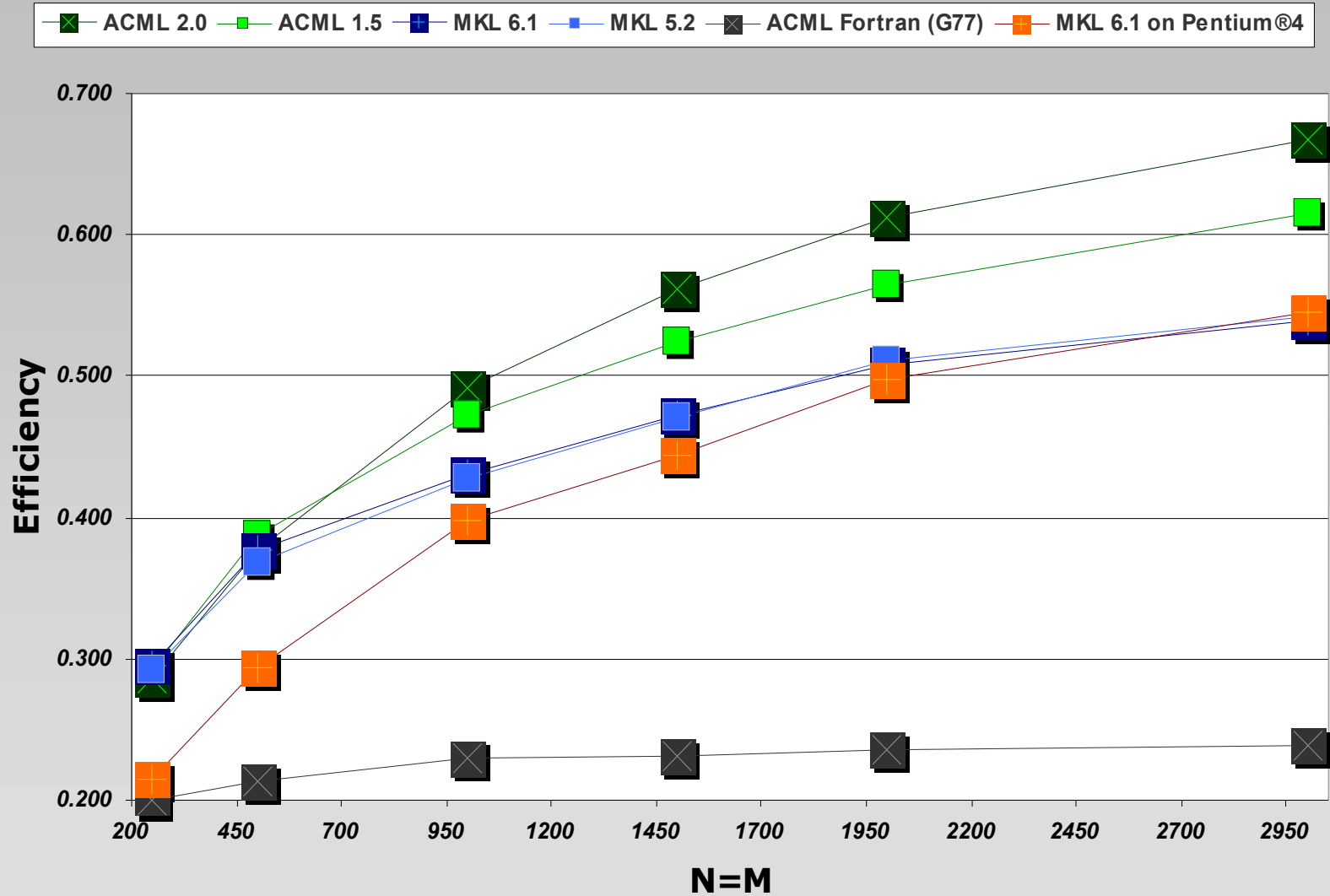
DGEMM (DP general matrix matrix multiply)



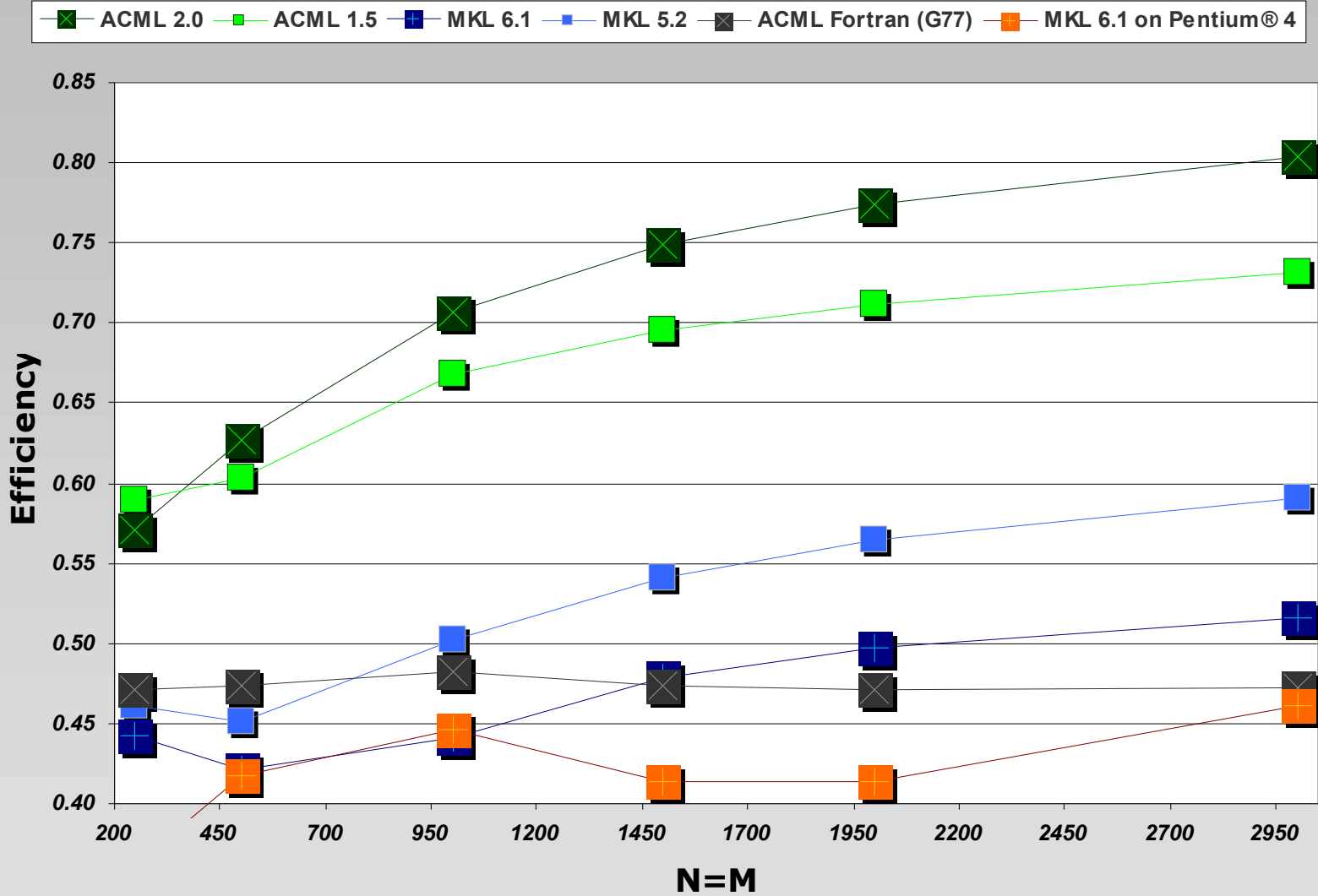
LAPACK

- Fortran-only, no assembly, but key routines are aggressively tuned
- Strong use of underlying BLAS ... building on underlying strength

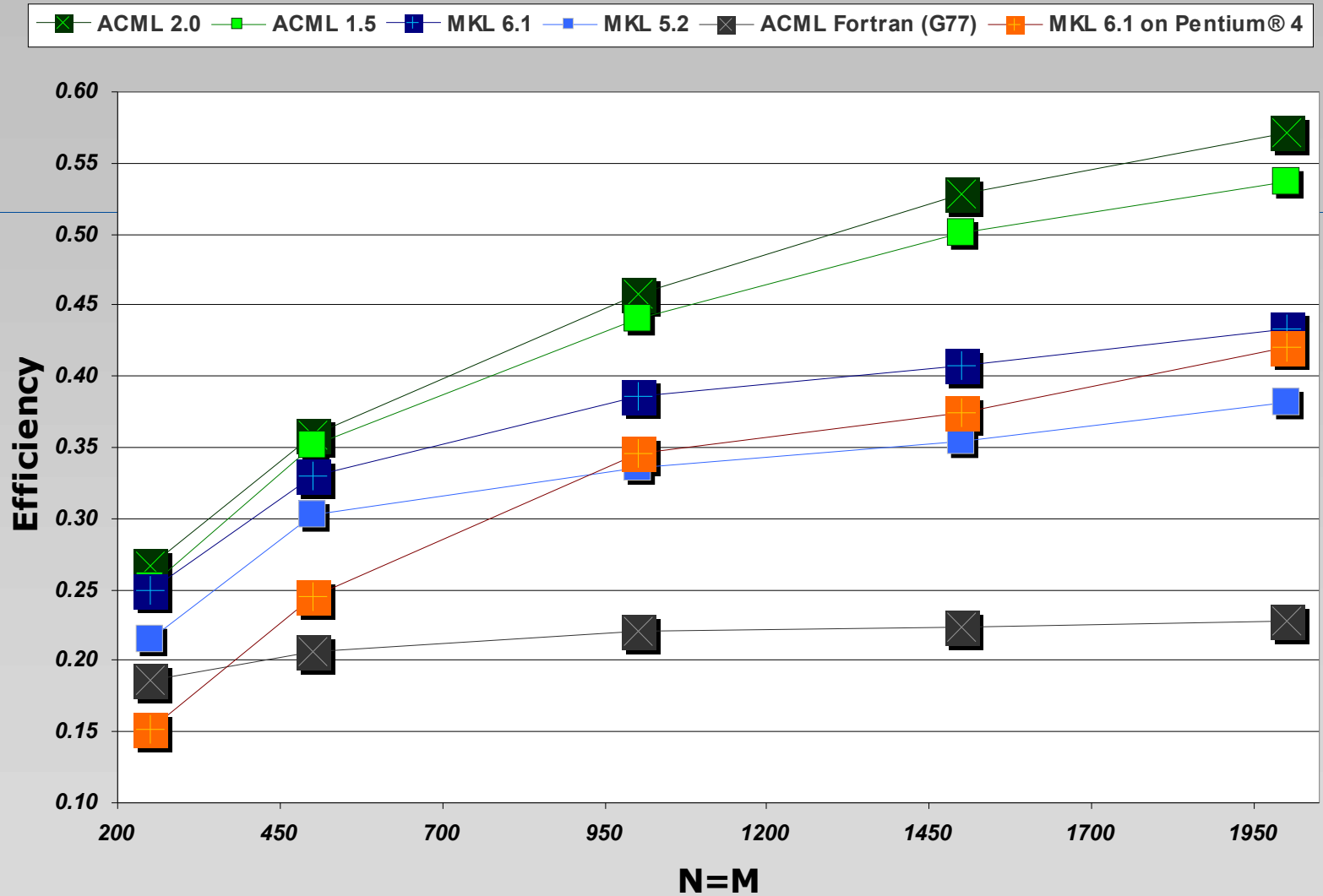
SGETRF (SP LU factorization)



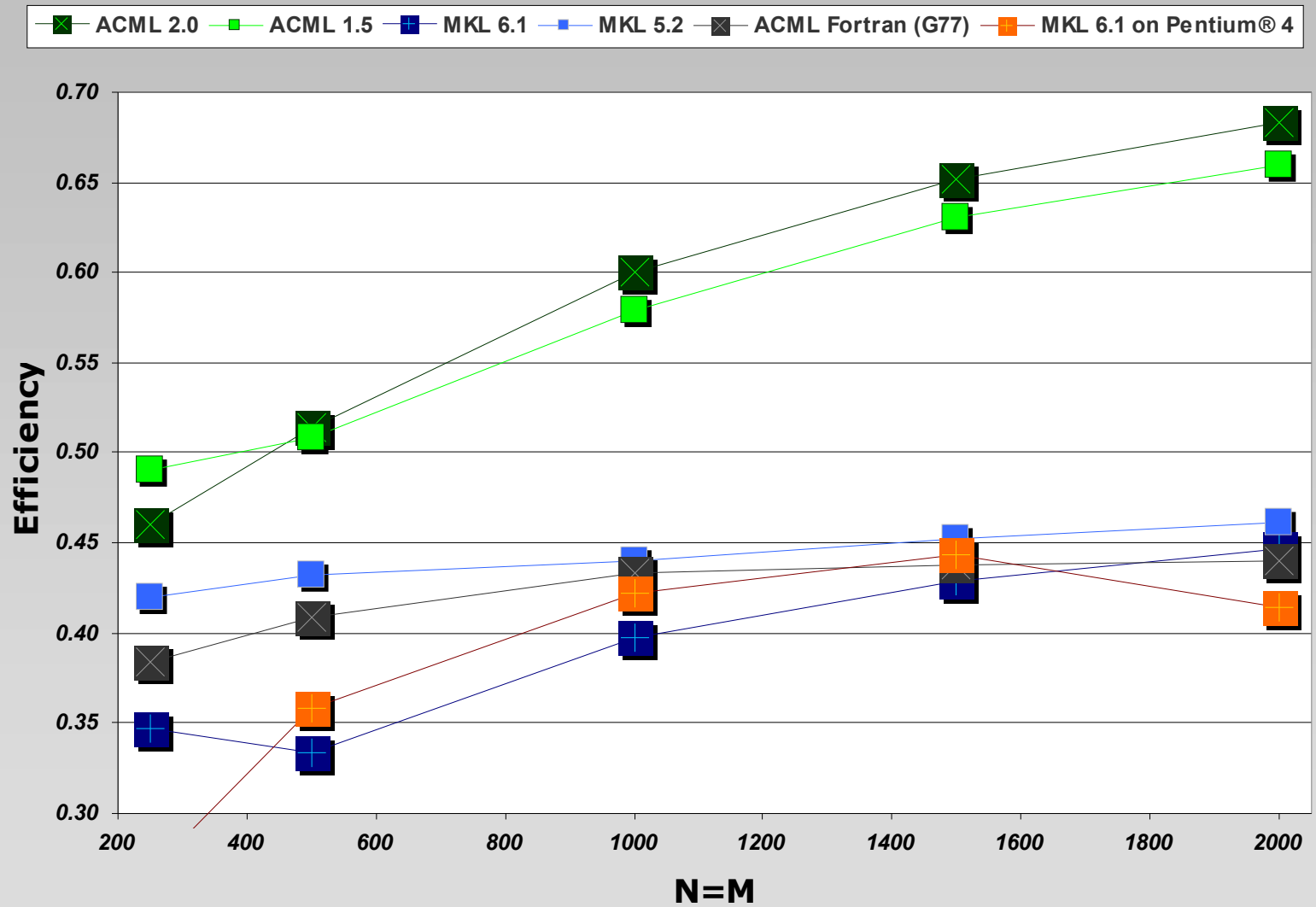
DGETRF (DP LU factorization)



SGEQRF (SP QR factorization)



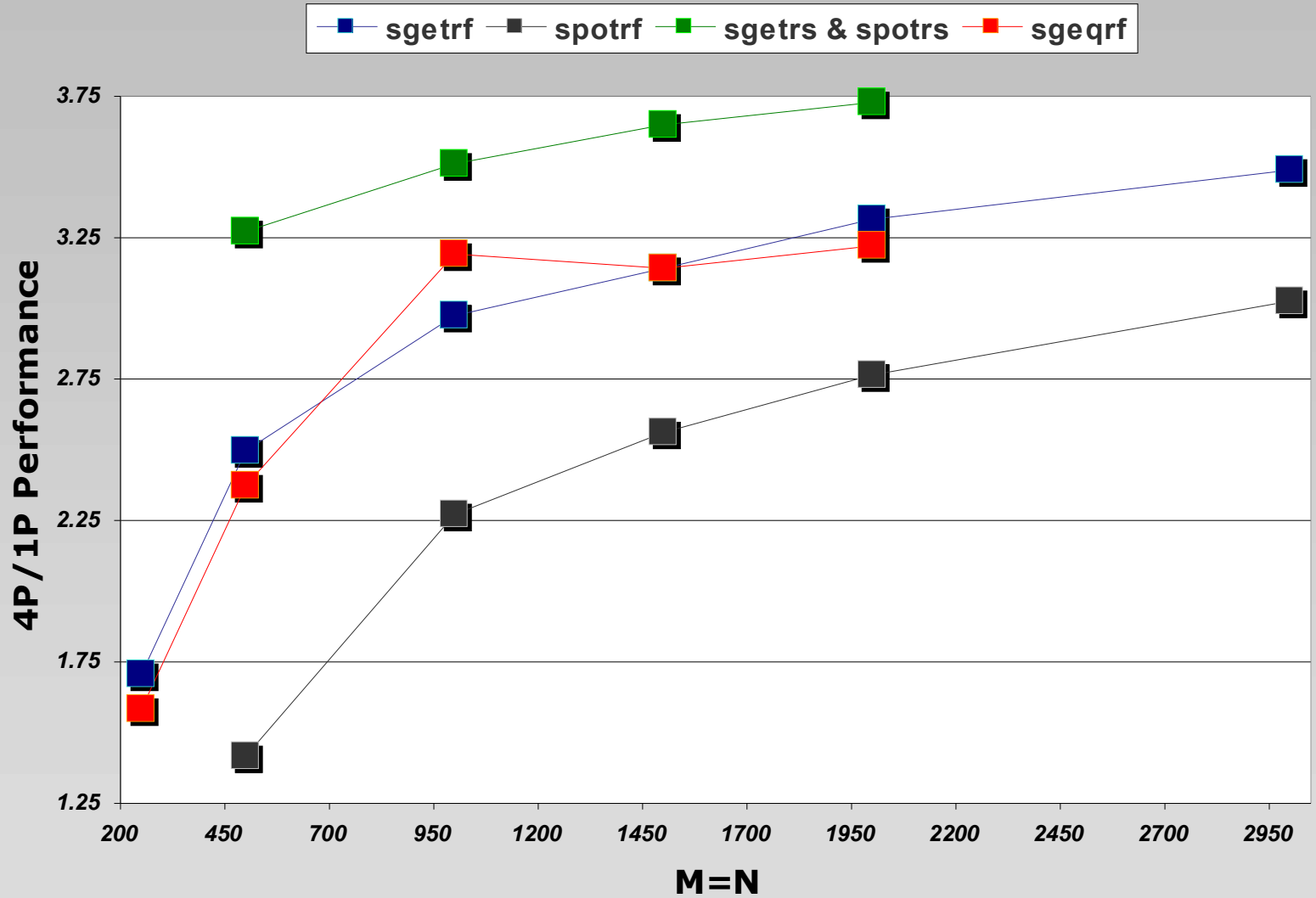
DGEQRF (DP QR factorization)



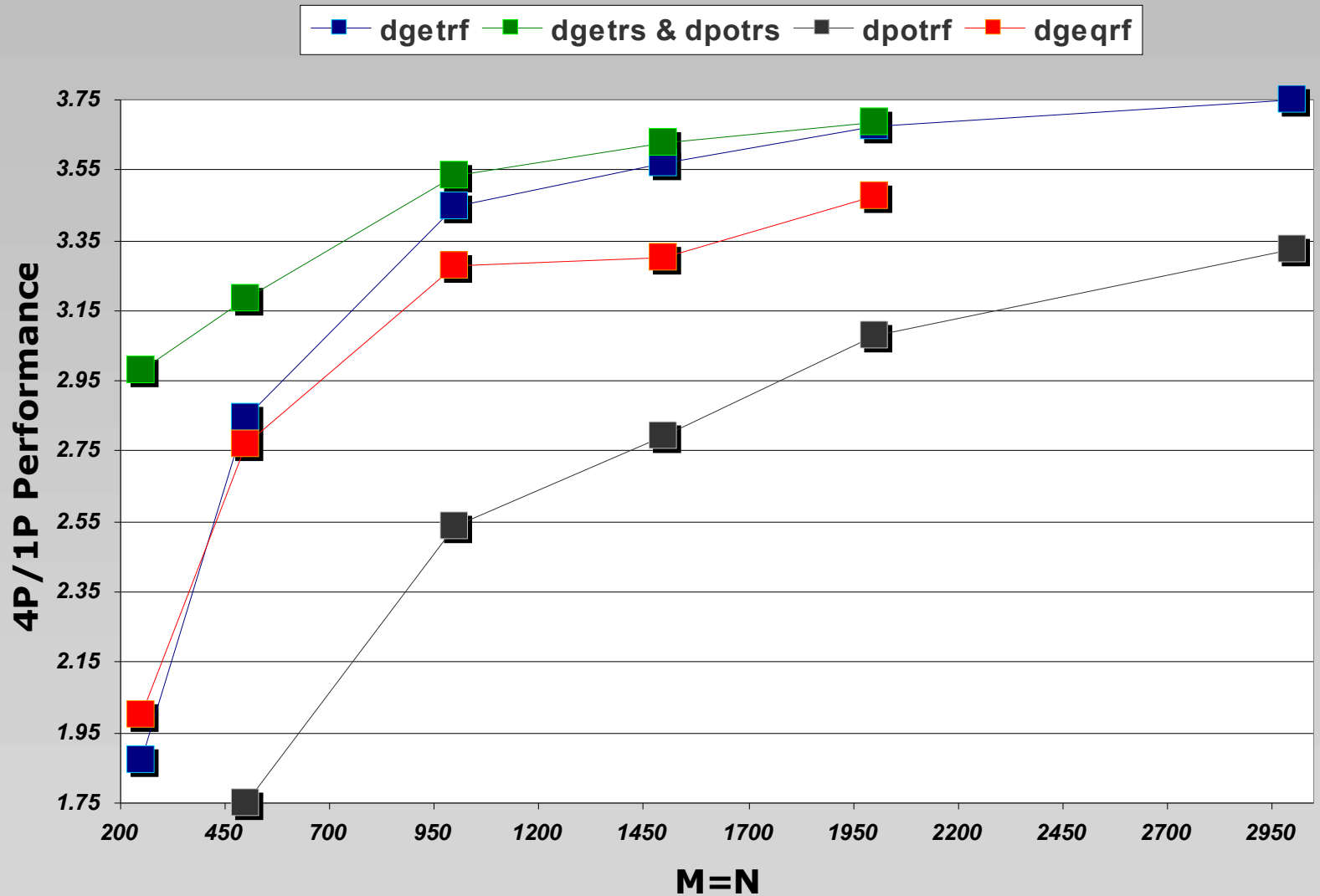
LAPACK multi-processor

- Key LAPACK routines are from NAG SMP library and use OpenMP. OpenMP switched on from ACML 1.5 once PGI compiler support.
- Other areas to gain multi-threading via OpenMP soon

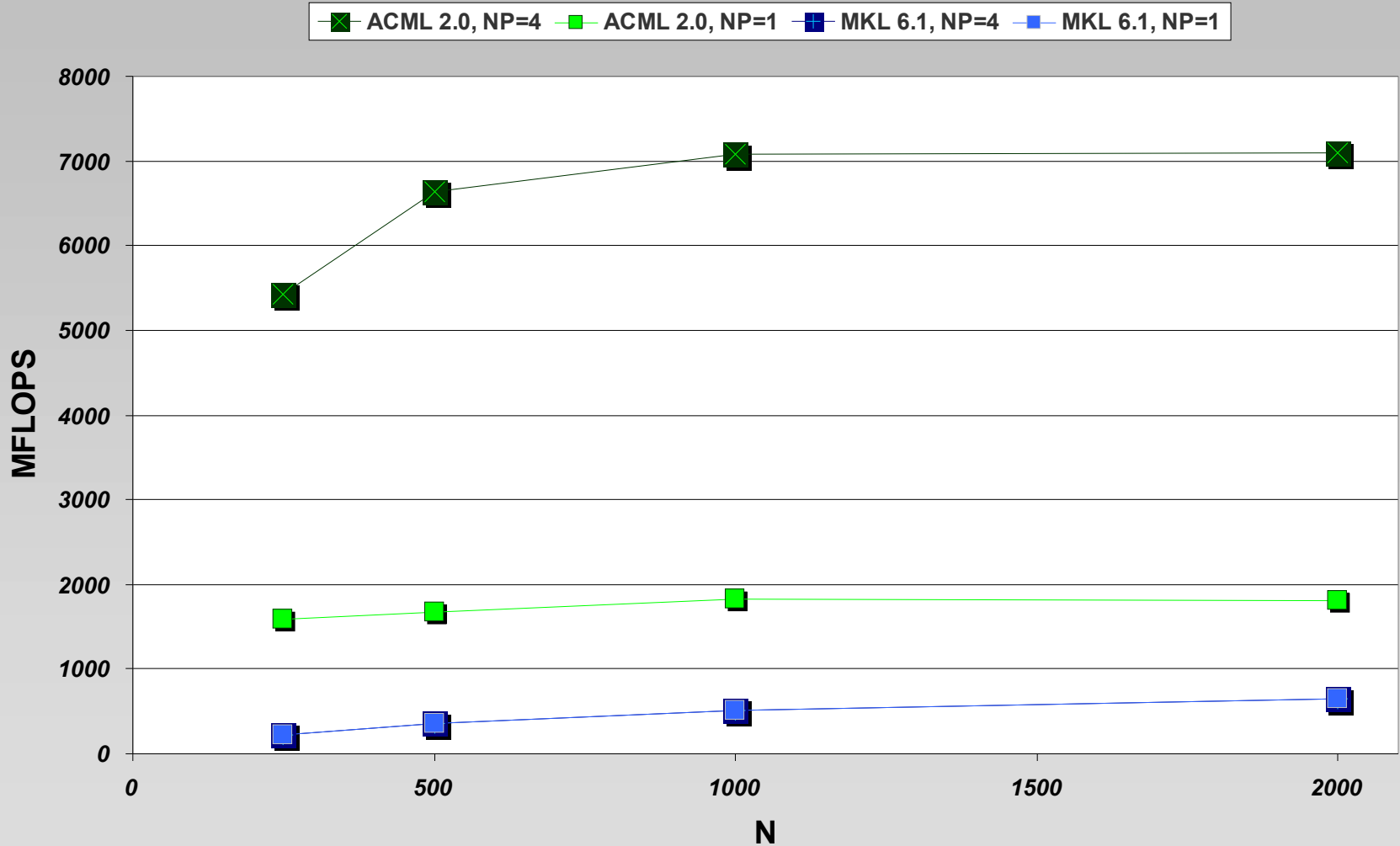
SP OpenMP key LAPACK routines scaling



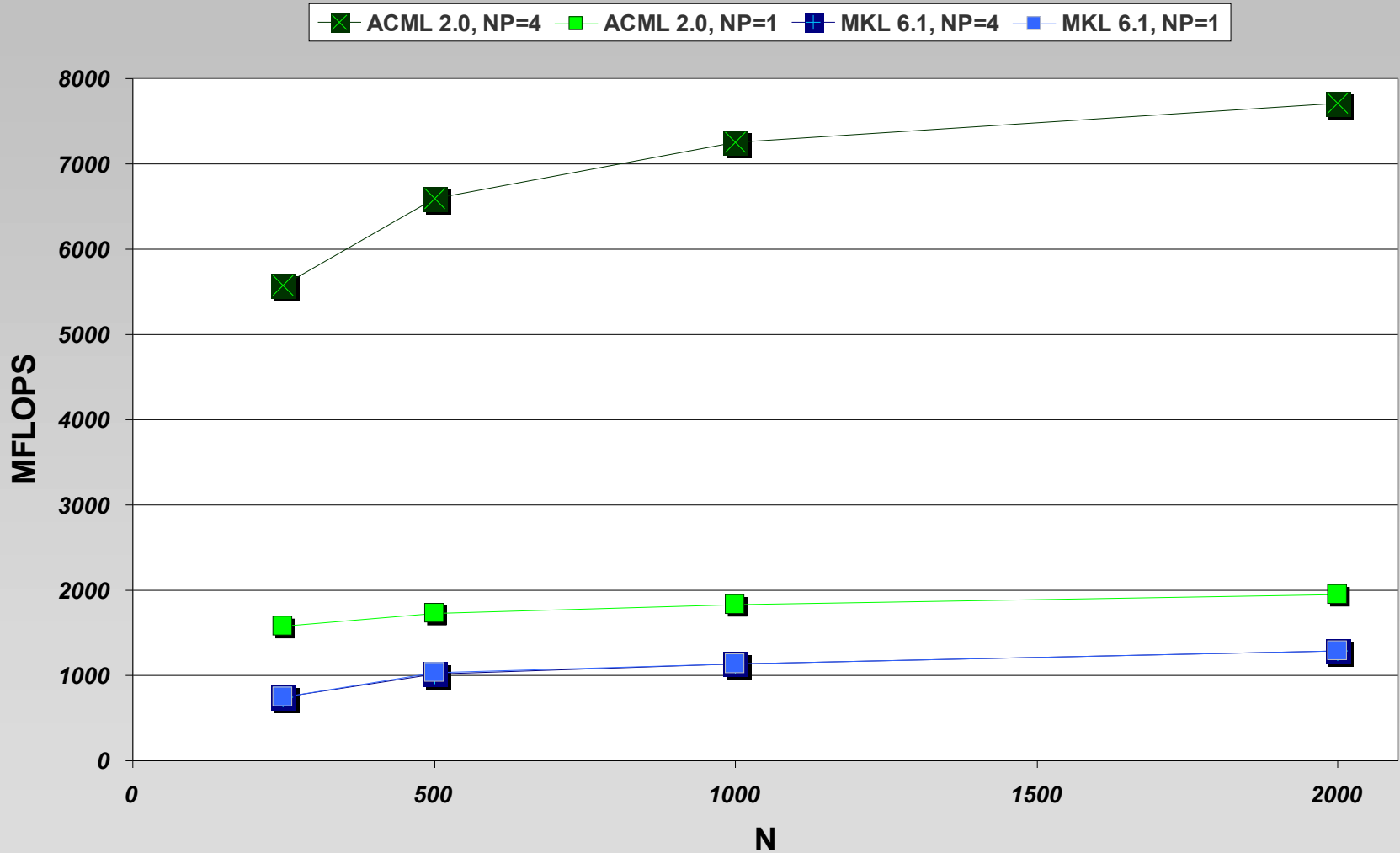
DP OpenMP key LAPACK routines scaling



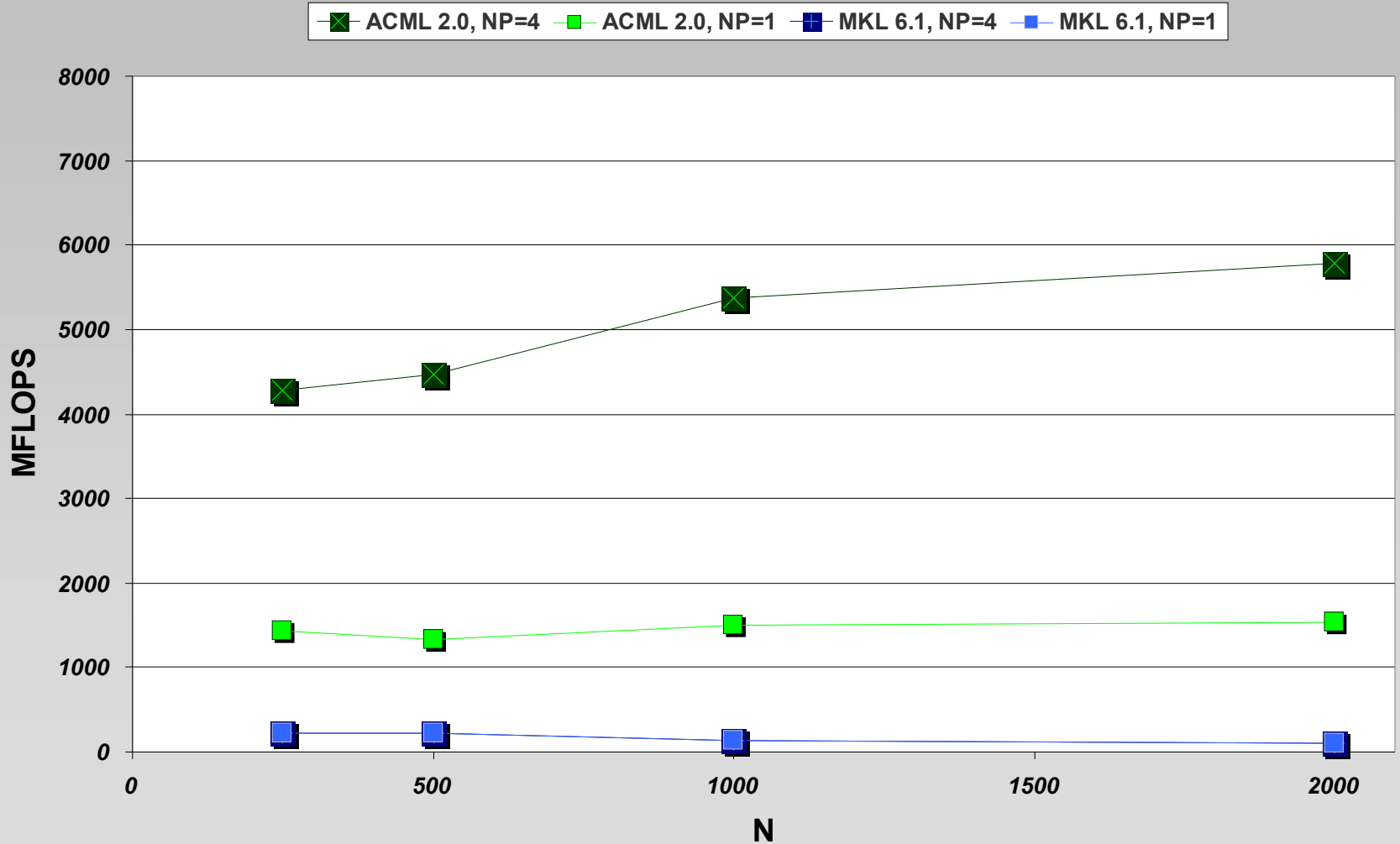
DSTEQR (DP Sym Eig)



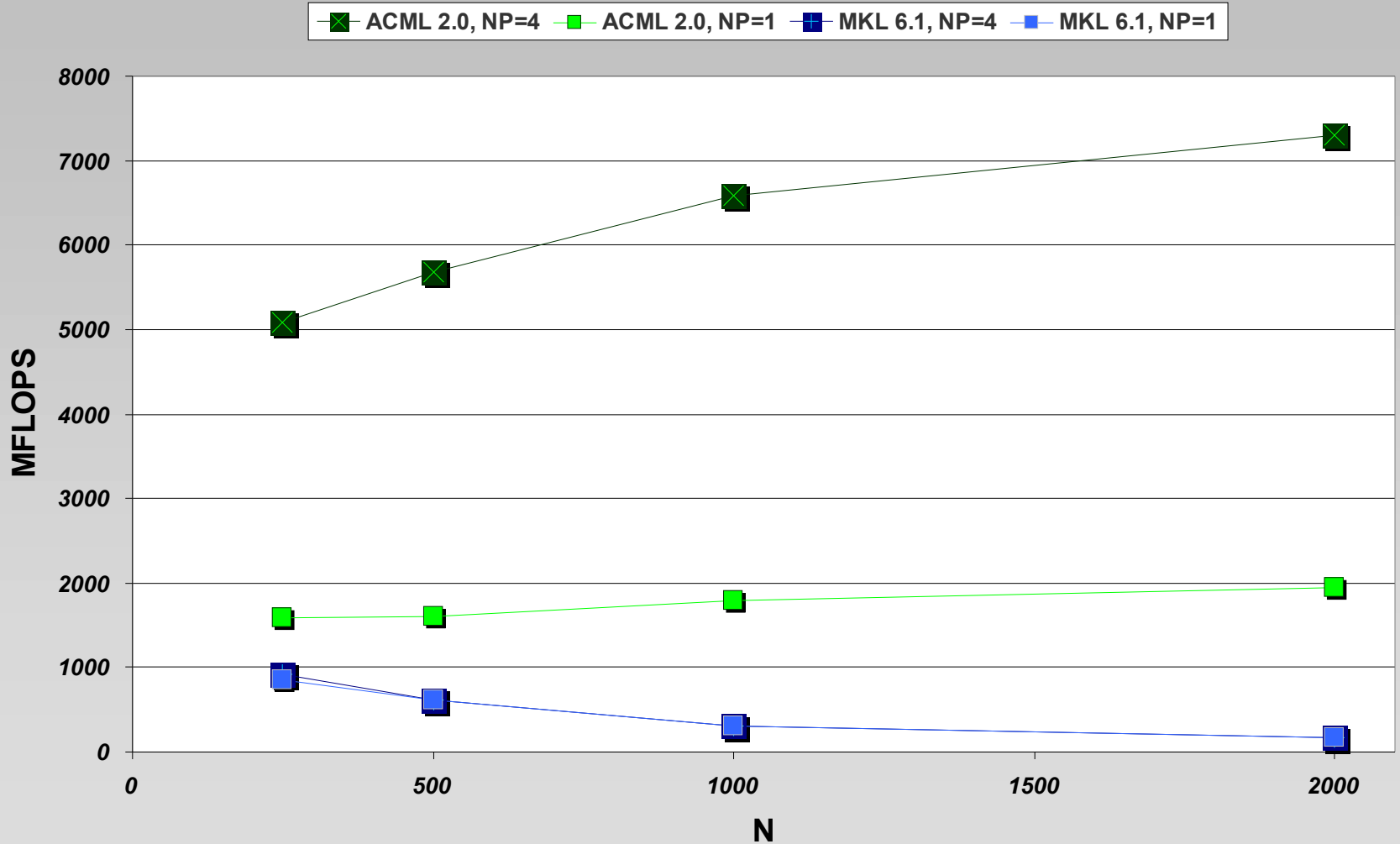
SSTEQR (SP Sym Eig)



DBDSQR (DP SVD)



SBDSQR (SP SVD)



Conclusions

- Math libraries are important for ongoing performance – and need to be accurate!
 - Q: How fast do you want the wrong answer?
 - A: Faster than the competition! ☹
 - High performance and accuracy are not always mutually exclusive!
- ISVs and customers are now driving further performance and functionality requirements ... just as it should be!
- Use of highly tuned assembly is sometimes critical:
 - For single precision code, Fortran compilers cannot compete with SSE2 (streaming SIMD) vectorized instructions.
 - But time consuming and difficult to maintain
- Healthy competition is good for everyone!

Over 30 years of numerical excellence



18/03 18/03 2KV6JTK
22/03 22/03 0VZEK4B

```
name: "Content-Type" content: "text/  
mime-version: "application/  
name: "application/  
name: "application/
```

Thank You

Over 30 years of numerical excellence

June 2004

Sun HPC Consortium, Heidelberg

32

NAG[®]