PARDISO - PARallel DIrect SOlver to solve SLAE on shared memory architectures

Solovev S. A, Pudov S.G

sergey.a.solovev@intel.com, sergey.g.pudov@intel.com

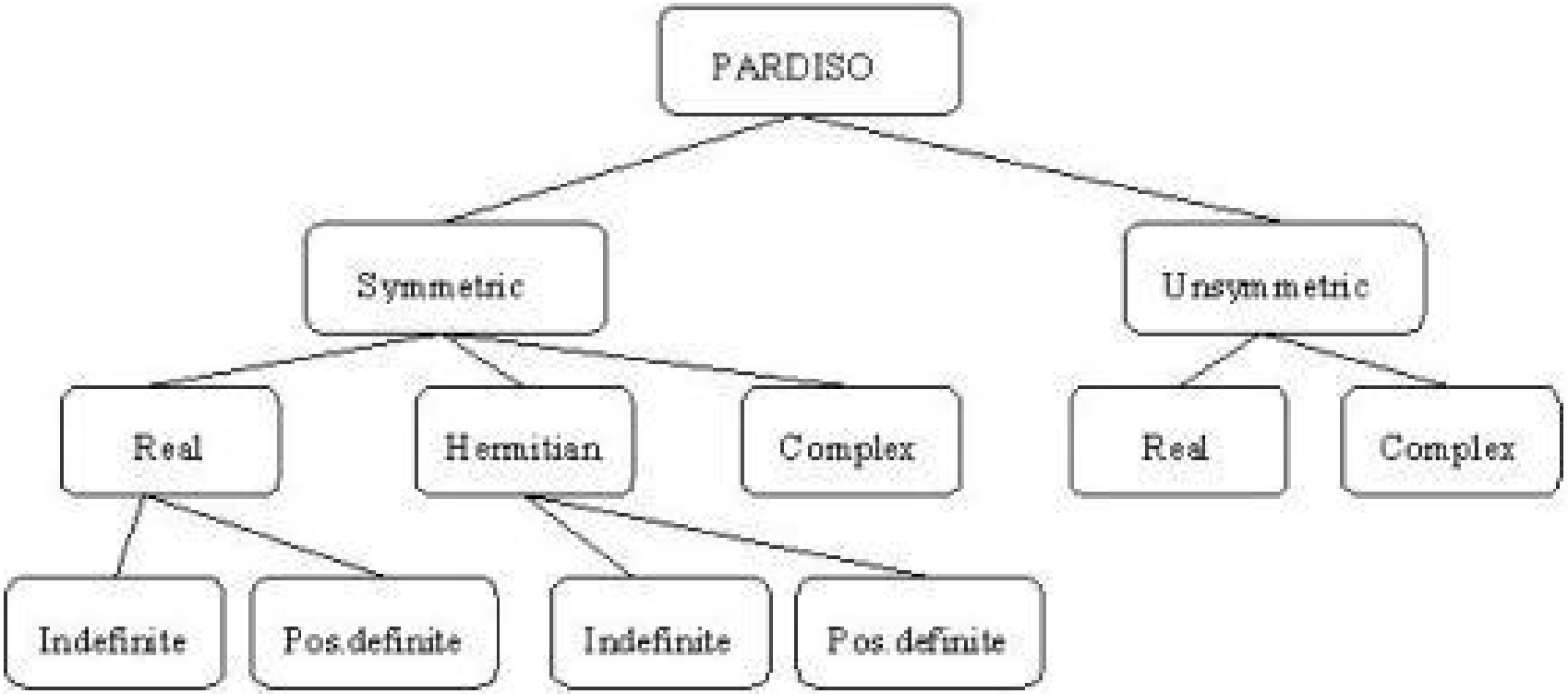
Technical Collateral Disclaimer

Agenda

- Introduction
- Main features of Intel MKL PARDISO
- Functionality of Intel MKL PARDISO
- Intel MKL PARDISO's competitors
- Memory using comparisons
- Performance comparisons
- Results of independent comparisons

Intel MKL PARDISO – Parallel Direct Sparse Solver Interface

Intel® Math Kernel Library (Intel® MKL) provides user-callable sparse solver software to solve real or complex, symmetric, structurally symmetric or non-symmetric, positive definite, indefinite or Hermitian sparse linear system of equations.



Main features of Intel MKL PARDISO

- Utilizes Intel MKL BLAS and LAPACK and uses shared-memory parallelism to improve numerical factorization performance.
- Solves wide class of SLAE as compared with iterative solvers.
- Minimizes RAM in use despite it is a direct solver.
- Up to linear speedup on multicore.
- Additional simplified interface DSS (Direct Sparse Solver) available from C and Fortran user codes.

Additional functionality of Intel MKL PARDISO

1. Reordering input matrix A. 3 options are available: choose one of two effective internal algorithms or provide specific reordering vector.
2. Built-in CG algorithm with preconditioner
3. Boosting ill-conditioned matrices
4. Iterative refinement
5. ILP64 interface (dealing with more than 2*10^9 elements).
6. Out-Of-Core version (intermediate arrays stored on disk)

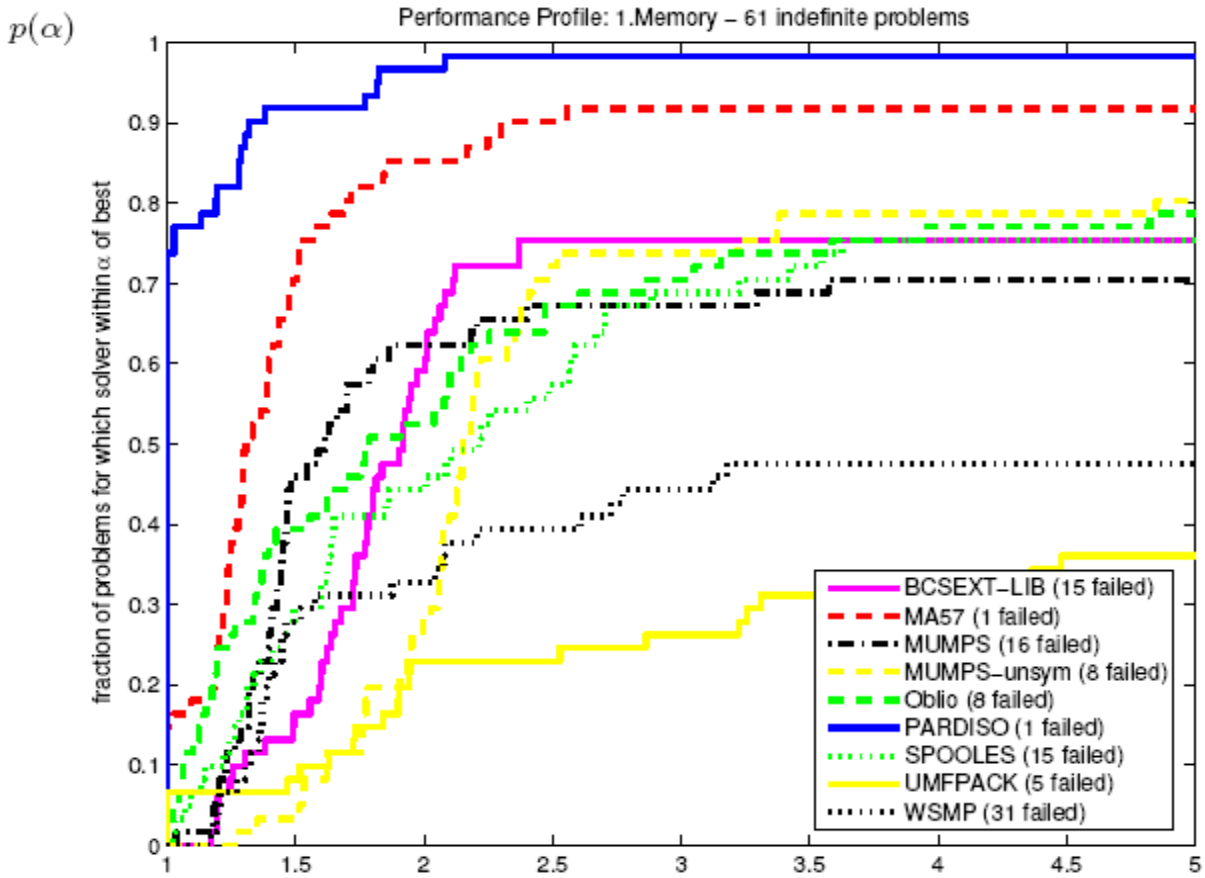
Competitors of Intel MKL PARDISO

3 major competitors of Intel MKL PARDISO: BCSLIB-EXT, MUMPS and TAUCS.

Code	Element entry	Scaling	Out-of-Core	Iterative refinement	Multiple rhs	Complex symmetric	Hermitian
BCSLIB-EXT	NO	NO	YES	X	YES	YES	YES
MA57	NO	YES	NO	YES	YES	YES	NO
MUMPS	YES	YES	YES	YES	NO	YES	NO
Oblio	NO	NO	YES	YES	YES	YES	NO
PARDISO	NO	YES	YES	YES	YES	YES	YES
SPOOLES	NO	NO	NO	NO	YES	YES	YES
SPRSBLKLLT	NO	NO	NO	NO	YES	NO	NO
TAUCS	NO	NO	YES	NO	NO	YES	YES
UMFPACK	NO	YES	NO	YES	NO	YES	YES
WSMP	NO	YES	NO	YES	YES	YES	YES

ftp:///ftp.numerical.rl.ac.uk/pub/reports/ghsRAL200505.pdf

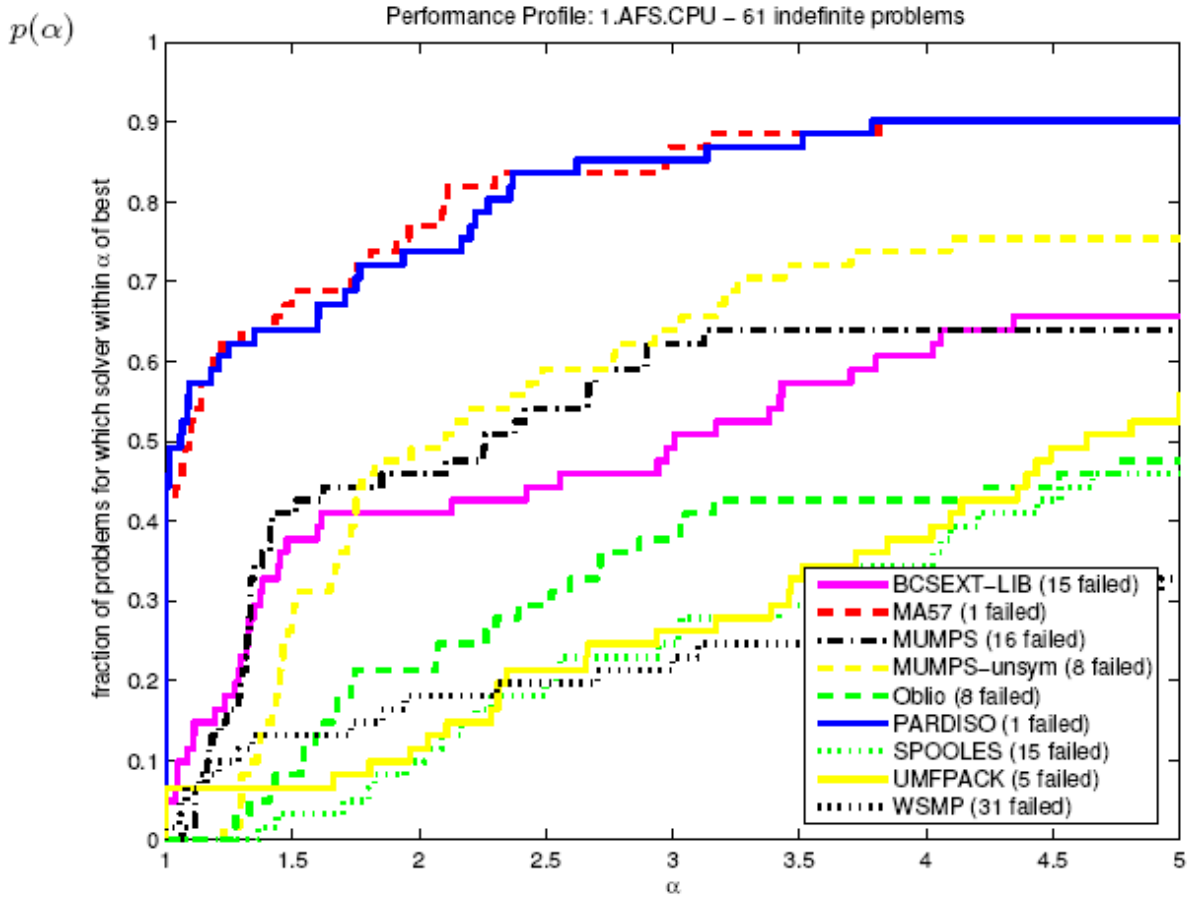
PARDISO Memory Utilization



This figure indicates a high correlation between the total memory used and the numbers of nonzeros in the factors …, with PARDISO requiring the least memory, followed by MA57 well above the rest

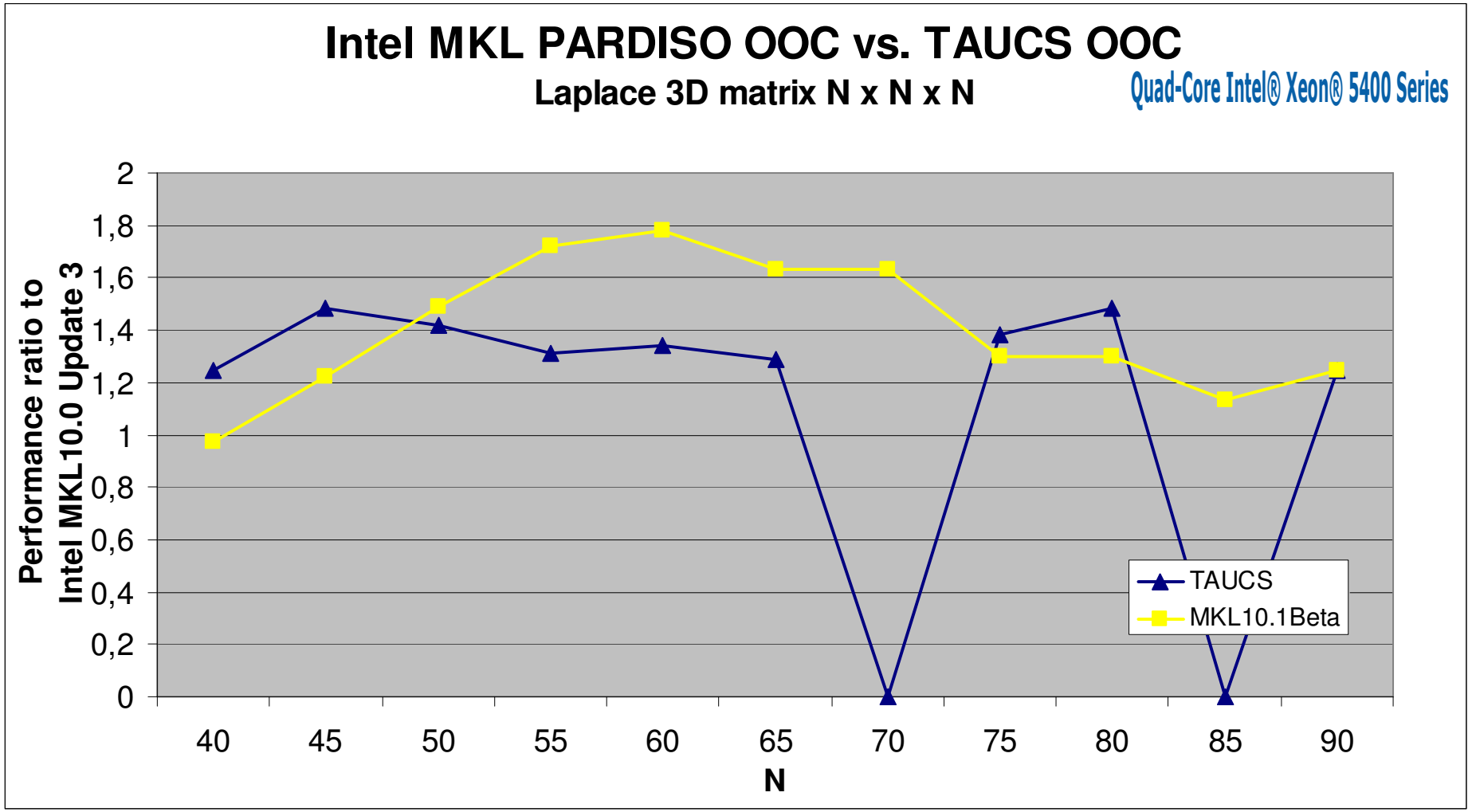
ftp://ftp.numerical.rl.ac.uk/pub/reports/ghsRAL200505.pdf

Performance profile, CPU time for the complete solution.

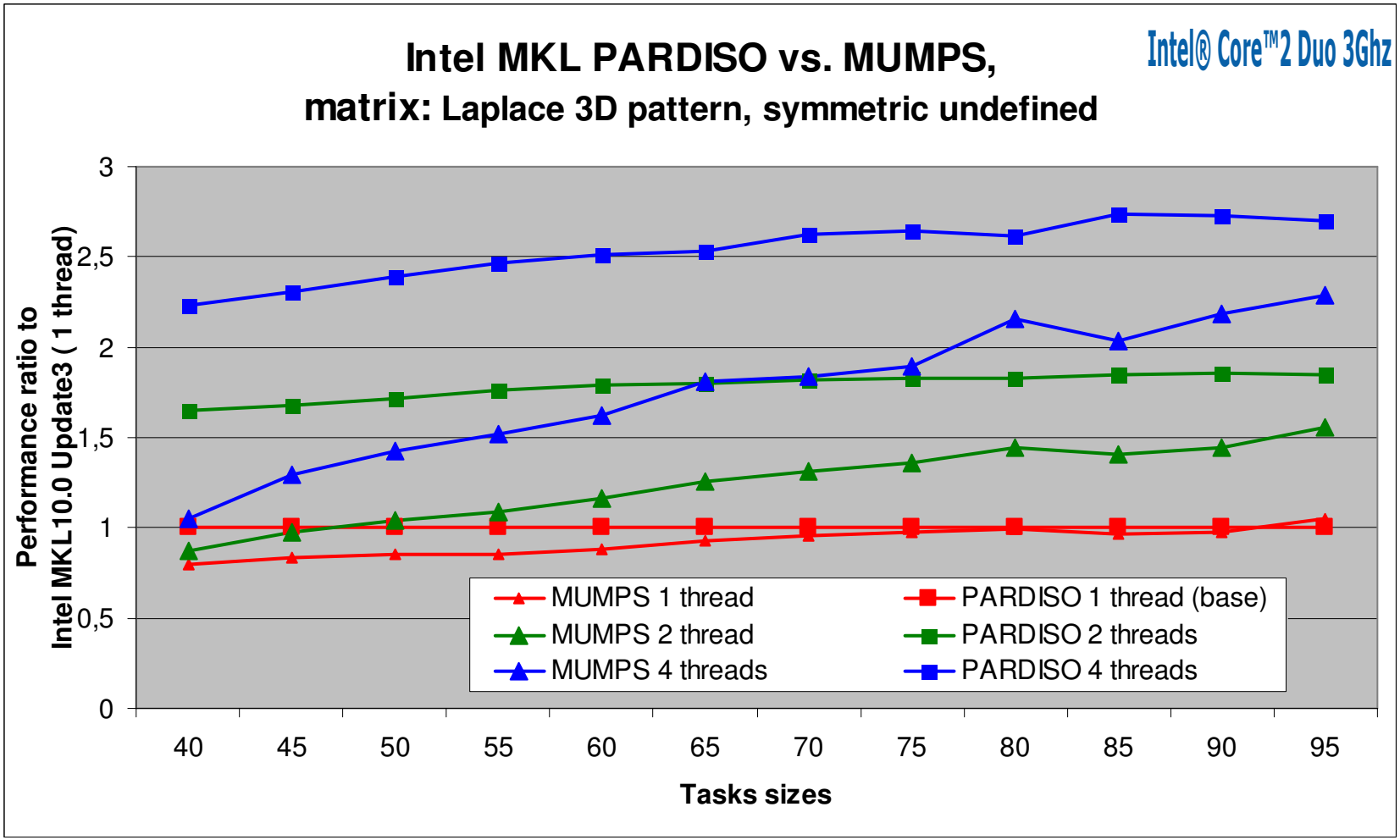


ftp://ftp.numerical.rl.ac.uk/pub/reports/ghsRAL200505.pdf

Performance comparison



Performance comparison



Performance tests and ratings are measured using specific computer systems and/or components and reflect the approximate performance of Intel products as measured by those tests. Any difference in system hardware or software design or configuration may affect actual performance. Buyers should consult other sources of information to evaluate the performance of systems or components they are considering purchasing. For more information on performance tests and on the performance of Intel products, visit Intel Performance Benchmark Limitations

Results of independent comparisons

- "Only three of the solvers (MA57, PARDISO and UMFPACK) had a success rate of 90% or better, while some of the others failed on 25% or more of the problems…"
- "Now examining the factorize times we see a significant gap between PARDISO and the remaining solvers…" (we see significant advantage of PARDISO over the remaining solver)
- "If many factorizations of matrices with identical sparsity patterns but differing values are required, WSMP and PARDISO are the strongest candidates, while if many solutions for a given matrix are needed BCSLIB-EXT, MA57 and PARDISO can be recommended…"
- "For indefinite problems, there are no strong stability guarantees without pivoting, and this is reinforced by the high percentage of failures for some algorithms in this case. The leading contenders here are MA57 and PARDISO…"
- "The careful use of static pivoting within PARDISO is surprisingly effective…"

ftp://ftp.numerical.rl.ac.uk/pub/reports/ghsRAL200505.pdf

Q&A