Karlsruhe Institute of Technology

# Power Consumption of Mixed Precision in the Iterative Solution of Sparse Linear Systems

H. Anzt, M. Castillo, J. C. Fernández, V. Heuveline,
R. Mayo, E. S. Quintana-Ortí, B. Rocker

No. 2011-01

Preprint Series of the Engineering Mathematics and Computing Lab (EMCL)


KIT – University of the State of Baden-Wuerttemberg and
National Research Center of the Helmholtz Association

www.emcl.kit.edu

Preprint Series of the Engineering Mathematics and Computing Lab (EMCL)
ISSN 2191–0693
No. 2011-01

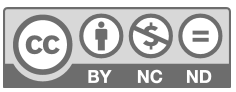
Impressum

Karlsruhe Institute of Technology (KIT)
Engineering Mathematics and Computing Lab (EMCL)

Fritz-Erler-Str. 23, building 01.86
76133 Karlsruhe
Germany

KIT – University of the State of Baden Wuerttemberg and
National Laboratory of the Helmholtz Association





# Power Consumption of Mixed Precision in the Iterative Solution of Sparse Linear Systems

Hartwig Anzt, Vincent Heuveline, Björn Rocker
*Institute for Applied and Numerical Mathematics 4*
*Karlsruhe Institute of Technology*
*Fritz-Erler-Str. 23, 76133 Karlsruhe, Germany*
{*hartwig.anzt,vincent.heuveline,bjorn.rocker*}*@kit.edu*

Maribel Castillo, Juan C. Fernández,
Rafael Mayo, Enrique S. Quintana-Ortí
*Depto. de Ingeniería y Ciencia de Computadores*
*Universidad Jaume I*
*12.071 - Castellón, Spain*
{*castillo,jfernand,mayo,quintana*}*@icc.uji.es*

***Abstract*—This paper presents a detailed analysis of a mixed precision iterative refinement solver applied to a linear system obtained from the 2D discretization of a fluid flow problem. The total execution time and energy need of different soft- and hardware-implementations are measured and compared with those of a plain GMRES-based solver in double precision. The time and energy consumption of individual parts of the algorithm are monitored as well, enabling a deeper insight and the possibility of optimizing the energy consumption of the code on a general-purpose multi-core architecture and systems accelerated by a graphics processor.**

***Keywords*-Energy Efficiency, Computational Fluid Dynamics (CFD), Large Sparse Linear Systems, Mixed Precision, Iterative Refinement, Hardware-Aware Computing, Energy Measurement**

## I. Introduction

### A. Motivation and contributions

The development of modern technology is characterized by simulations, that are no longer performed in physical experiments, but in mathematical modeling and numerical computations. At the same time, the enormous energy consumption to perform these simulations is becoming an increasing problem. Today, already after short time, the energy cost of running a supercomputer often exceeds the hardware acquisition cost [1]. Especially in the context of Exascale Computing, the energy consumption and the Carbon Footprint are major challenges addressed by an increasing number of researchers [2].

In the simulation of real-world phenomena, solving large-scale sparse linear systems is often an essential part, demanding for immense computational capacities as well as the usage of algorithms optimized for the specific applications. In many cases, a combination of different arithmetic precisions in the solvers may trigger an acceleration of the application without sacrificing the accuracy of the final result [3], [4].

In this paper, we show how a mixed precision error correction method [5] for solving sparse linear systems of equations can be adapted to a specific hardware platform, yielding significant gains in the computation time and energy demands. As an secondary contribution of the paper, we analyze the correlation between computation time and energy needs for different parameters controlling the solver, which allows the investigation of a trade-off between accuracy, time and energy.

### B. Related work

Iterative refinement is a well-known technique to improve the quality of the computed solution to a linear system $Ax = b$ [6]. Mixed precision in combination with iterative refinement has been reported as an efficient means to reduce the execution time of dense linear systems solvers [7]–[10] as well as sparse linear system solvers [11], [12]. However, none of the previous works considers the impact of this combination on energy.

The idea of applying a mixed precision iterative refinement variant instead of a plain solver to reduce power demands was recently addressed in [13], [14]. The first work performs a "potential" analysis of the benefits of this technique by considering only the theoretical power consumption of the different hardware components obtained from vendor specifications. In this paper we provide much stronger experimental evidence of the energy savings by using actual power measurements. Alike ours, the second work also employs real power measures; the target architecture considered there, though, did not include a hardware accelerator and the iterative solver (CG) also differs.

### C. Paper Organization

In the next section, we offer the necessary mathematical background. There we outline the rationale of iterative refinement methods (section II-A); introduce the technique of using different floating-point precision formats within these algorithms (section II-B); and review the underlying mathematical problem arising from a CFD application (section II-C). The following section describes the hardware setup of the test platform (section III-A) and some implementation issues (section III-B).

In section IV, we perform several experiments, analyzing the computation time and energy consumption of different

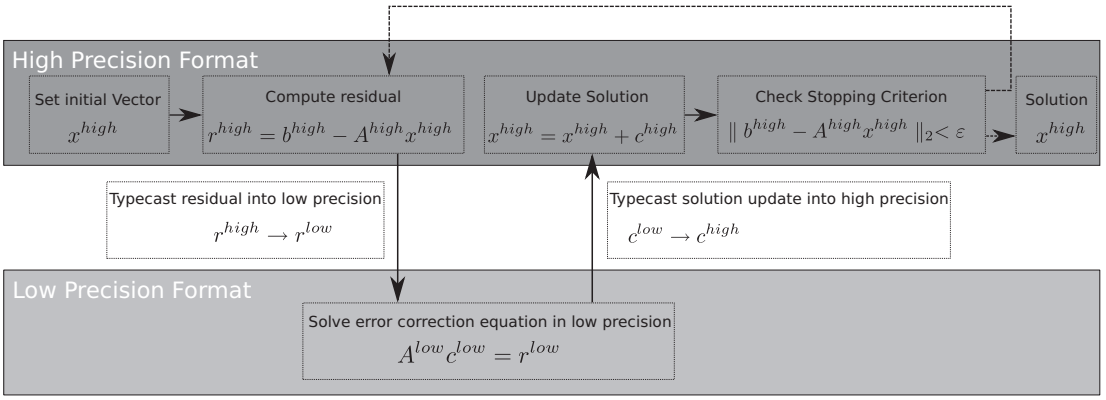



Figure 1. Stages of the mixed precision approach to an iterative refinement solver.

software setups on a current general-purpose multi-core processor (or CPU). The study also includes a hybrid multi-core+graphics processor (GPU) platform, which demonstrates the potential of GPUs to accelerate scientific codes, but also to deliver a better energy-per-arithmetic-operation ratio. The global study shows the benefits of using mixed precision iterative refinement implementations of linear solvers on both computation time and power consumption. At the same time, the detailed energy analysis reveals the possible optimizations with respect to the used hardware resources. In the last section, we offer a few concluding remarks and give a brief overview about the potential of future implementations on hybrid hardware technology.

## II. Mathematical Background

### A. Iterative Refinement

The motivation for the iterative refinement method can be obtained from Newton's method. Consider a function $f$ and let $x_i$ denote the solution in the $i$-th step:

$$x_{i+1} = x_i - (\nabla f(x_i))^{-1} f(x_i). \quad (1)$$

This method can be applied to the function $f(x) = b - Ax$ with $\nabla f(x) = A$, where $Ax = b$ is the linear system that has to be solved.

By defining the residual $r_i := b - Ax_i$, one then obtains

$$\begin{aligned} x_{i+1} &= x_i - (\nabla f(x_i))^{-1} f(x_i) \\ &= x_i + A^{-1}(b - Ax_i) \\ &= x_i + A^{-1} r_i. \end{aligned}$$

Denoting the solution update with $c_i := A^{-1} r_i$, and using an initial guess $x_0$ as the starting value, an iterative algorithm can be defined, where *any linear solver* can be used as error correction solver. (In this paper we will use GMRES –see section III-B3, [15], [16]– without preconditioning, to solve the general sparse linear system associated with the CFD application that is analyzed.)

At each iteration, the inner correction solver searches for a vector $c_i$ such that $Ac_i = r_i$ is approximated, and the solution approximation is updated by $x_{i+1} = x_i + c_i$.

### B. Mixed Precision Iterative Refinement

The idea underlying mixed precision iterative refinement methods is to use different precision formats within the algorithm, updating the solution approximation in high precision, but computing the error correction term in lower precision.

A comparison of the mixed precision iterative refinement solver (see, e.g., Figure 1) with a plain solver easily reveals that the iterative refinement method has more computations to execute. In particular, each outer loop of the mixed precision solver consists of the computation of the residual error term, a typecast into low precision, a vector update, the scaling process, the inner solver for the correction term, the transformation of the data back into high precision, and the solution update. The computation of the residual error itself consists of a matrix-vector multiplication, a vector addition, and a scalar product. Furthermore, when a hybrid CPU-GPU architecture is used to accelerate the computations, a certain amount of data has to be transmitted between the host and the device memory address spaces. On the other hand, the mixed precision iterative refinement approach can potentially outperform a plain solver in high precision if the additional computations and typecasts are compensated by the much faster operation pace of low precision arithmetic in hardware accelerators like, e.g., GPUs.

If the final accuracy does not exceed the smallest number $\epsilon_{\text{low}}$ that can be represented in the lower precision, the mixed precision solver computes exactly the same solution approximation as if the solver was performed in the high precision format. Theoretically, any precision can be chosen, but in most cases it is convenient to use the IEEE 754 standard formats.

When an iterative method is employed as error correction solver (i.e., to solve $Ac_i = r_i$), the iterative approaches to the Krylov subspace methods are specially appealing, as these

provide an approximation of the residual error iteratively in every computation loop. Hence, one is able to set a certain relative residual stopping criterion for the iterative error correction solver. Possible Krylov subspace solvers include the CG algorithm, GMRES, BiCGStab, etc. [15]. The mixed precision iterative refinement method based on a certain error correction solver poses the same demands to the linear problem as the Krylov subspace solver employed within it.

It should be mentioned, that the solution update of the mixed precision iterative refinement solver is usually sub-optimal for the outer system, since the discretization of the problem in the lower precision format contains rounding errors and, therefore, it solves a perturbed problem.

### C. Mathematical Application

As the test problem, we employ a linear system of equations that was obtained from a 2D discretization of the Navier-Stokes Equations modeling the fluid flow in a Venturi Nozzle. The sparsity pattern of the coefficient matrix is captured in Figure 2. The problem size (dimensions of the square matrix $A$) is $n$=395,009 and the number of nonzeros $nnz$=3,544,321.

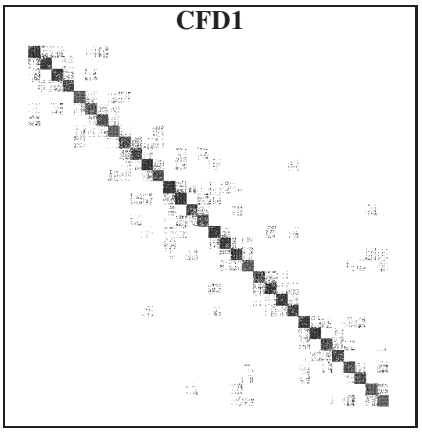



Figure 2. Sparsity plots showing the nonzero-structure of the CFD test-matrices.

## III. Test Configuration

### A. Hardware Platform and Energy Measurement

All experiments reported in this paper were performed on a platform consisting of two Intel Xeon E5410 Quad-Core processors at 2.33 GHz, with 4 GB of RAM, connected via PCIe (16x) to an NVIDIA Tesla C1060 board.

Power is measured at two different points. A commercial external power meter (Watts Up Pro .net) samples power for the global system once per second (1 Hz). Given the low resolution of the measures, and the "noise" introduced by hardware components as the disk or the network interface card as well as the inefficiencies of the power supply, we added an alternative sampling point, with a higher resolution, using an internal power meter. This is an ASIC operating at a frequency of 25 Hz (25 samples per second) which is composed of a number of resistors connected in series with the power source: thus, the drop of power voltage across the series yields a direct estimation of the power being consumed. In this case, we attach the internal power meter to the lines connecting directly the power supply unit with the GPU and the motherboard (chipset plus processors) so as to obtain the energy consumption of the computing hardware. Samples from both the external and the internal power meter are collected in a separate system; thus, the measurements do not affect the performance of the tests. Figure 3 illustrates the connection between the target platform and the energy measurement hardware.

### B. Implementation Issues

*1) Libraries:* Our CPU-implementations of the mixed precision iterative refinement solver as well as the plain GMRES solver operate on vectors using the BLAS (*basic linear algebra subprograms*) functionality provided in Intel Math Kernel Library [17] (MKL version 11.1). Although MKL also includes an implementation of the sparse matrix-vector multiplication, invoked from within the GMRES algorithm and the residual computation (the test matrix is sparse, stored in the CRS format [18]), we decided not to use it, as our experiments reported the superiority of a plain implementation of the sparse matrix-vector kernel for the specific structure and dimensions of matrix CFD1. In order to improve performance, the CPU code was compiled using the Intel `icc` compiler (version 11.1) with the flags `-O3 -parallel -ipo` [19], which enable aggressive optimizations and parallel execution using multiple threads on the Intel multi-core processor.

When the error corrector solver within the mixed precision iterative refinement implementation solver is executed on the GPU, the vector operations are performed with the respective CUBLAS routines in [20] (version 2.2). NVIDIA `nvcc` compiler (version 2.2) with an up-to-date CUDA driver was employed in the GPU. The CUBLAS routines are used whenever possible; a kernel for sparse matrix-vector multiplication was implemented following the guidelines suggested in [21].

*2) Memory Management:* For the CPU-implementation of the plain GMRES in double precision, the matrix is stored in double precision only. The mixed precision iterative refinement method requires an additional copy in single precision. In the GPU-implementation, this additional copy is stored in the GPU memory during the initialization, such that no regular memory-copy from host to device is necessary during the solution stage.

*3) GMRES Solver:* For both the plain solver in double precision as well as the error correction solver in the iterative refinement framework, we use a GMRES algorithm (*Generalized Minimum Residual Method*). GMRES is a

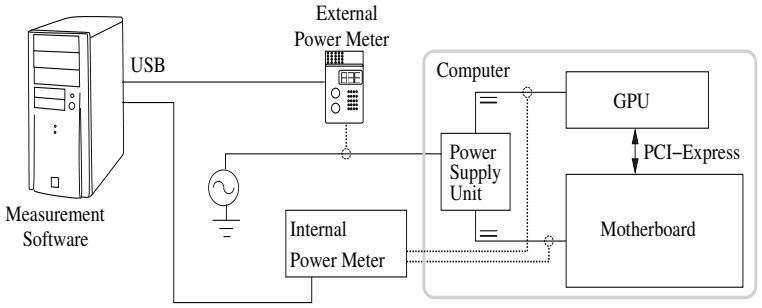



Figure 3. Hardware platform and sampling points.

projection method that operates on Krylov subspaces $V_m = \mathcal{K}_m(A, v) = \text{span}\{v, Av, A^2v, \ldots, A^mv\}$, generated by the Arnoldi algorithm. It was designed for the solution of linear systems where the coefficient matrix $A$ is neither necessarily symmetric nor positive definite [15], [16]. Indeed, GMRES also works for non symmetric semi-positive definite systems, and is especially appropriate for large-scale sparse matrices.

As GMRES uses the Arnoldi algorithm to generate the Krylov subspace, and the entire Krylov subspace $\mathcal{K}_n$ spans $\mathbb{R}^n$, in exact arithmetic and after $n$ steps, GMRES computes the exact result to a linear system of dimension $n$. Therefore, GMRES is in fact a direct method, like other Krylov subspace solvers, that computes the analytically exact solution in $n$ steps. In practice, for large linear systems, difficulties appear in the method due to a linear increase in computational and storage costs, and to the loss of orthogonality of the Krylov subspaces triggered by rounding errors. Because choosing a small number $m \ll n$ of iterations often yields a good approximation of the result, one usually employs GMRES as an iterative solver, with a stopping criterion depending on the residual norm.

In the plain GMRES algorithm, the whole Krylov basis has to be stored until the residual has reached a certain threshold. Therefore, for large linear systems, the memory and computational costs of this method become prohibitive. To avoid this, a variant known as RESTART-GMRES, or GMRES-($m$), is often used, where the Krylov subspace and the approximation is not computed until the residual has reached the demanded threshold, but restarted after a certain number of steps ($m$).

The advantages of the restarted algorithm are that the orthogonality of the computed Krylov subspaces is preserved to a higher degree due to the restart of the Krylov-subspace generator and the computational and memory costs are decreased, as the linear problem stays at a lower dimension, and only $m$ Krylov subspace vectors have to be stored (see Algorithm 1).

*4) Solver Parameters:* While we use a plain GMRES algorithm in restart-variant as reference solver, the mixed precision iterative refinement method uses Restart-GMRES solver in low precision as error correction solver. For the different tests, we set the restart parameter to $m$=30. Furthermore, in most of our experiments we fix the relative residual stopping criterion for the final solution approximation to $10^{-10}$. Due to the iterative residual computation in the case of the plain GMRES solvers, the mixed GMRES solvers based on the mixed precision iterative refinement method usually yield a more accurate approximation, since they compute the residual error explicitly. However, as the difference is generally small, the results can be compared.

In the first tests, we vary the relative residual stopping criterion $\varepsilon_{\text{inner}}$ of the error correction solver inside the mixed precision iterative refinement solver. In all other tests, when analyzing the energy consumption of the individual parts of the solver and the comparison to the plain solver implementation, we set the inner stopping criterion to $10^{-1}$, since this choice is the optimal for our application from the points of view of execution time and energy needs.

## IV. Numerical Tests

In the first experiment, we analyze the effect of the inner stopping criterion on the execution time and the energy consumption of different hardware implementations of the mixed precision iterative refinement method. To compare with a direct algorithm without iterative refinement, we also consider a plain GMRES solver, operating in double-precision arithmetic, and using the same relative residual stopping criterion of $10^{-10}$ for the approximation to the final solution.

Figure 4 gathers the results of this first experiment on CFD1. For the execution time as well as for the energy consumption, we observe the superiority of the mixed precision iterative refinement solver (label "IT. REF.") using the very loose inner stopping criterion $\varepsilon_{\text{inner}} = 0.1$. Considering the CPU implementations (bars corresponding to 1, 2 and 4 threads), the plain solver in double precision (label "GMRES") runs almost two times longer, while the internal powermeter measures energy savings larger than $3\times$. Due to the noise introduced by other hardware components, the external powermeter offers less detailed values and will be discarded it in the following experiments. The mixed

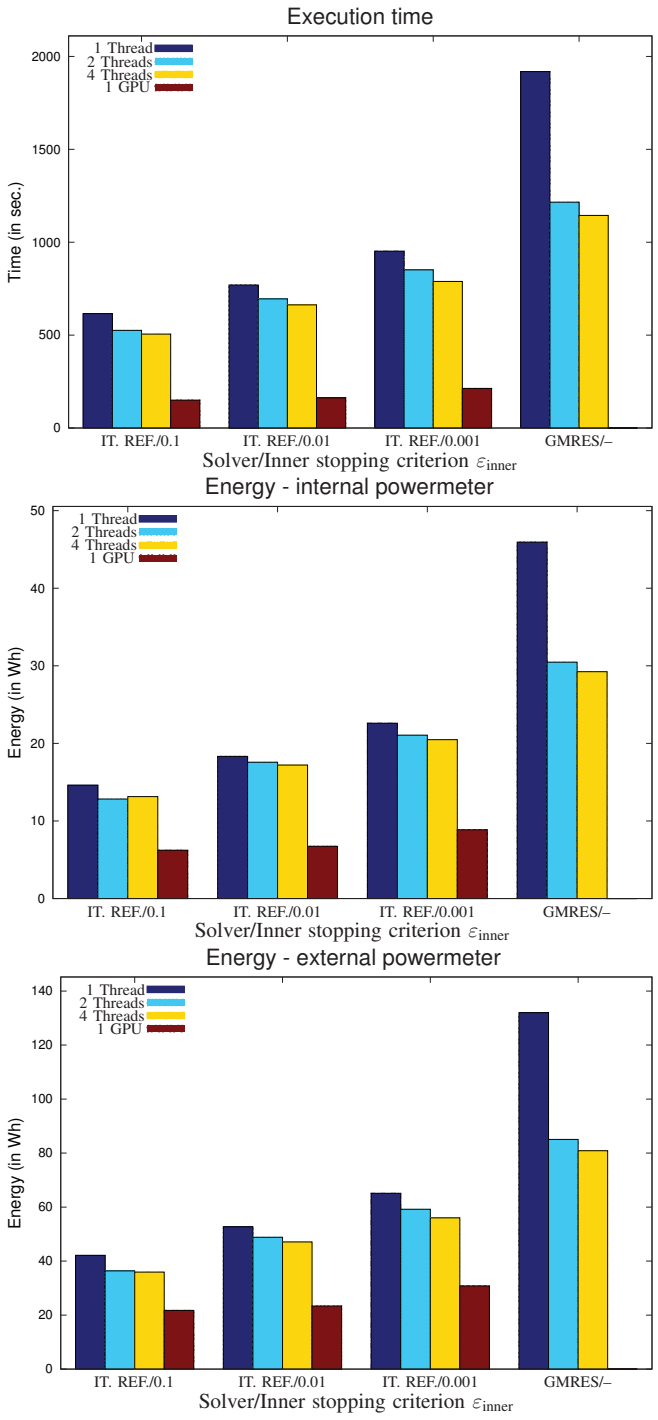



Figure 4. Impact of the inner stopping criterion on the performance of the unsymmetric sparse linear system solvers on CFD1. Top: execution time; center: energy consumption measured with the internal powermeter; bottom: energy consumption measured with the external powermeter.

1: **for** $(l = 1, l++)$ **do**
2: Compute $r_0 = b - Ax_0$, $d_0 = \beta_0 = \| r_0 \|_2$, $v_1 = \frac{r_0}{\beta_0}$
3: **for** $(j = 1, j \leq m, j++)$ **do**
4: % Iteration process of GMRES
5: Compute $w_j = Av_j$
6: **for** $(i = 1, i \leq j, i++)$ **do**
7: % Arnoldi's method
8: $h_{i,j} = \langle w_j, v_i \rangle$
9: $w_j = w_j - h_{ij} v_i$
10: **end for**
11: $\omega = \| w_j \|_2$
12: **for** $(i = 1, i < j, i++)$ **do**
13: % Apply former rotation to $h_k$
14: $\tilde{h} = c_i h_{i,j} + s_i h_{i+1,j}$
15: $h_{i+1,j} = -s_i h_{i,j} + c_i h_{i+1,j}$
16: $h_{i,j} = \tilde{h}$
17: **end for**
18: **if** $(\omega \leq |h_{j,j}|)$ **then**
19: % Compute new rotation
20: $t_j = \frac{\omega}{|h_{j,j}|}$
21: $c_j = \frac{h_{j,j}}{|h_{j,j}|\sqrt{1+t_j^2}}$
22: $s_j = \frac{t_j}{\sqrt{1+t_j^2}}$
23: **else**
24: $t_j = \frac{h_{j,j}}{\omega}$
25: $c_j = \frac{t_j}{\sqrt{1+t_j^2}}$
26: $s_j = \frac{1}{\sqrt{1+t_j^2}}$
27: **end if**
28: $h_{j,j} = c_j h_{j,j} + s_j \omega$ % Apply rotation to rest of $\hat{H}$
29: $d_j = -s_j d_{j-1}$ %Apply the rotation to the RHS
30: $d_{j-1} = c_j d_{j-1}$
31: **end for**
32: solve $H_l y = d$ with the Gauss-Algorithm
33: Define the matrix $V_l = [v_1 \dots v_l]$
34: Compute the approximation $x_l = x_0 + V_l y$
35: **if** $(|d_l| \leq \varepsilon)$ **then**
36: stop
37: **end if**
38: **end for**

Algorithm 1: GMRES-$(m)$ Algorithm.

precision iterative refinement using the GPU as coprocessor (bar corresponding to GPU) for the error correction solver yields a speedup of 12 in computation time compared with the sequential implementation of the plain solver, and almost 8 compared with the implementation using 4 cores/threads. Still, the energy savings are lower, since running the GPU is expensive, especially because it is not switched off but remains active even when not needed. The internal powermeter measures energy savings of a factor around 6 for the implementation using 4 cores, while the external powermeter only gives a factor of 4. This is again due to the noise

generated by the power supply unit and other hardware components.

The purpose of the next experiment is to analyze the computation time and the power consumption of the individual parts of one single run of the mixed precision iterative refinement solver. From the insights gained from the previous study, we set the inner stopping criterion to $\varepsilon_{\text{inner}} = 0.1$. We split the algorithm into the following parts:

- T1 **Reading matrix**. Read data from file.
- T2 **Initialization**. Allocate and initialize matrix and vectors, in double as well as in single precision. In case the GPU is used as a coprocessor, the single precision matrix and the vectors are also allocated and initialized in the GPU memory. This involves, among other tasks, the transfer of the full matrix contents from the CPU memory to the GPU memory.
- T3 **Typecast double→single**. The residual is typecasted from double to single precision. When the GPU is used as coprocessor, the typecasted residual is also copied to the device.
- T4 **Error correction solver**. The error correction equation is solved in single precision. If the GPU is involved, this task is mainly executed by it.
- T5 **Typecast single→ double**. The solution update obtained from the error correction solver is typecasted from single precision to double precision. When operating with the GPU, this task also includes transferring the data from device back to host.
- T6 **Solution update**. Using the solution update, the double precision solution approximation is improved.
- T7 **Residual computation**. The new residual is computed and the relative residual stopping criterion is checked. If it is fulfilled, the algorithm stops; otherwise, it cycles back to the typecast for the residual T3.

Note that T1 and T2 are not strictly part of a single run of the mixed precision solver, but are performed before the iteration commences; we include their values for completeness. The results for tasks T3–T7 are averaged by the number of iterations of the mixed precision solver.

Table I shows the average time/energy consumption for the individual tasks T1–T7 in one iteration loop of the iterative refinement method applied to the matrix associated with CFD1. As one could expect, the error correction solver is by far the most demanding part of the algorithm, not only in computation time but also in power input.

When the mixed precision iterative refinement method is run on the hybrid hardware platform consisting of CPU and GPU, the GPU runs on idle all time when the operations are exclusively performed by the CPU. This concerns especially tasks T1, T6 and T7. A more efficient system would shut down the GPU while not using it (assuming the cost of shutting down/powering up is close to negligible). The same is true vice-versa. While in case of the GPU-implementation, T4 is mainly executed by the GPU, the CPU still runs at high frequency. Here, technologies like dynamic voltage and frequency scaling (DVFS) may become very handy, as they enable to lower the processor frequency while not being used.

| Execution time | | | | |
|---|---|---|---|---|
| Task | 1 thread | 2 threads | 4 threads | 1 GPU |
| T1 | 2.53e+00 | 2.65e+00 | 2.60e+00 | 2.48e+00 |
| T2 | 3.72e-02 | 3.68e-02 | 3.70e-02 | 1.22e+00 |
| T3 | 3.49e-03 | 3.60e-03 | 3.60e-03 | 5.20e-03 |
| T4 | 5.10e+01 | 4.35e+01 | 4.18e+01 | 1.21e+01 |
| T5 | 3.14e-03 | 2.80e-03 | 2.87e-03 | 4.62e-03 |
| T6 | 3.55e-03 | 3.04e-03 | 2.68e-03 | 3.11e-03 |
| T7 | 2.75e-02 | 2.65e-02 | 2.57e-02 | 2.69e-02 |

| Energy – internal powermeter | | | | |
|---|---|---|---|---|
| Task | 1 thread | 2 threads | 4 threads | 1 GPU |
| T1 | 5.03e-02 | 5.12e-02 | 5.11e-02 | 8.51e-02 |
| T2 | 9.07e-04 | 6.86e-04 | 7.47e-04 | 3.72e-02 |
| T3 | 6.10e-05 | 0.00e+00 | 1.24e-04 | 2.24e-04 |
| T4 | 1.21e+00 | 1.06e+00 | 1.09e+00 | 5.08e-01 |
| T5 | 5.40e-05 | 0.00e+00 | 0.00e+00 | 2.84e-04 |
| T6 | 6.48e-05 | 6.48e-05 | 9.79e-05 | 0.00e+00 |
| T7 | 5.36e-04 | 6.68e-04 | 6.14e-04 | 8.86e-04 |

Table I
DETAILED ANALYSIS OF INDIVIDUAL TASKS OF THE MIXED PRECISION ITERATIVE REFINEMENT SOLVER ON CFD1. TOP: TIME (IN SEC.); BOTTOM: ENERGY MEASURED WITH THE INTERNAL POWERMETER (IN WH).

Our last experiments evaluates the trade-offs between execution time, accuracy and power consumption. There we again set the inner stopping criterion $\varepsilon_{\text{inner}} = 0.1$ for the mixed precision solvers. We then analyze the execution time and energy necessary to reach a certain accuracy of the solution. Figure 5 shows an homogeneous increase of both factors, linear with the magnitude of the relative residual of the computed solution. The figure illustrates the large gaps in execution and power demand of the mixed precision solvers compared with those of the plain GMRES solver. Furthermore, we observe that except for the plain GMRES implementation, the usage of multiple cores does not offer significant benefits to the solving process. In contrast, good acceleration is provided by the GPU attached to the system used for the low precision arithmetic computations of the mixed precision iterative refinement method.

## V. CONCLUSIONS AND FUTURE WORK

The experiments show the high potential of using a mixed precision iterative refinement method for the test case we considered. At the same time, the efficient use of hybrid hardware systems triggers considerable energy savings. These can even be enlarged by using dynamic voltage and frequency control for the processors, and hardware systems where coprocessors like GPUs only run on demand. Future work will include these topics as well as more advanced ones like, e.g., the design of an intelligent runtime that aids in choosing hardware resources and solver type at execution

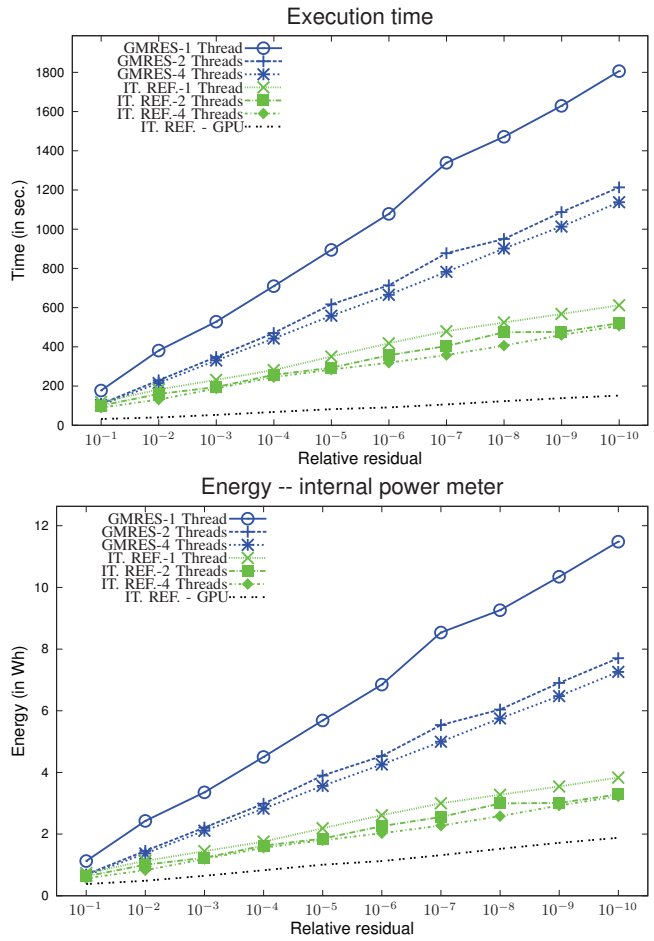



Figure 5. Impact of the outer stopping criterion on the performance of the unsymmetric sparse linear system solvers on CFD1. Top: execution time; bottom: energy consumption measured with the internal powermeter.

time. From the numerical point of view, future work will also consider the impact of preconditioning and reordering techniques on the performance of the sparse linear system solvers.

## Acknowledgments

The authors would like to thank V. Roca and G. Fabregat, for their technical support with the energy measurement framework.

The authors from the Universidad Jaume I were supported by project CICYT TIN2008-06570-C04-01 and FEDER.

## References

[1] F. Lampe, *Green-IT, Virtualisierung und Thin Clients : Mit neuen IT-Technologien Energieeffizienz erreichen, die Umwelt schonen und Kosten sparen*. Vieweg + Teubner, 2010.

[2] *ExaScale Computing Study: Technology Challenges in Achieving Exascale Systems*. DARPA/IPTO, 2008.

[3] H. Anzt, B. Rocker, and V. Heuveline, "An Error Correction Solver for Linear Systems: Evaluation of Mixed Precision Implementations," EMCL Preprint Series, 2010. [Online]. Available: http://www.emcl.kit.edu/preprints/emcl-preprint-2010-01.pdf

[4] H. Anzt, T. Hahn, V. Heuveline, and B. Rocker, "GPU Accelerated Scientific Computing: Evaluation of the NVIDIA Fermi Architecture; Elementary Kernels and Linear Solvers," EMCL Preprint Series, 2010. [Online]. Available: http://www.emcl.kit.edu/preprints/emcl-preprint-2010-04.pdf

[5] H. Anzt, B. Rocker, and V. Heuveline, "Mixed Precision Error Correction Methods for Linear Systems: Convergence Analysis based on Krylov Subspace Methods," EMCL Preprint Series, 2010. [Online]. Available: http://www.emcl.kit.edu/preprints/emcl-preprint-2010-02.pdf

[6] N. J. Higham, *Accuracy and Stability of Numerical Algorithms*, 2nd ed., 2002.

[7] M. Baboulin, A. Buttari, J. J. Dongarra, J. Langou, J. Langou, P. Luszcek, J. Kurzak, and S. Tomov, "Accelerating scientific computations with mixed precision algorithms," *Computer Physics Communications*, vol. 180, no. 12, pp. 2526–2533, 2009.

[8] A. Buttari, J. J. Dongarra, J. Langou, J. Langou, P. Luszcek, and J. Kurzak, "Mixed precision iterative refinement techniques for the solution of dense linear systems," *Int. J. of High Performance Computing & Applications*, vol. 21, no. 4, pp. 457–486, 2007.

[9] S. Barrachina, M. Castillo, F. D. Igual, R. Mayo, and E. S. Quintana-Ortí, "Solving dense linear systems on graphics processors," in *Proceedings of the 14th international Euro-Par conference on Parallel Processing*, ser. Lecture Notes in Computer Science, 5168, E. Luque, T. Margalef, and D. Benítez, Eds. Springer, 2008, pp. 739–748.

[10] S. Barrachina, M. Castillo, F. D. Igual, R. Mayo, E. S. Quintana-Ortí, and G. Quintana-Ortí, "Exploiting the capabilities of modern GPUs for dense matrix computations," *Concurrency and Computation: Practice and Experience*, vol. 21, no. 18, pp. 2457–2477, 2009.

[11] D. Göddeke, R. Strzodka, and S. Turek, "Performance and accuracy of hardware–oriented native–, emulated– and mixed–precision solvers in FEM simulations," *Int. J. of Parallel, Emergent and Distributed Systems*, vol. 22, no. 4, pp. 221–256, 2007.

[12] D. Göddeke and R. Strzodka, "Performance and accuracy of hardware–oriented native–, emulated– and mixed–precision solvers in FEM simulations (part 2: Double precision GPUs)," Fakultät für Mathematik, TU Dortmund, Tech. Rep., July 2008, ergebnisberichte des Instituts für Angewandte Mathematik, Nummer 370.

[13] H. Anzt, B. Rocker, and V. Heuveline, "Energy efficiency of mixed precision iterative refinement methods using hybrid hardware platforms," *Computer Science - Research and Development*, vol. 25, Issue 3, pp. 141–149, 2010.

[14] C. Bekas and A. Curioni, “A new energy aware performance metric,” *Computer Science - Research and Development*, vol. 25, Issue 3, pp. 187–195, 2010.

[15] Y. Saad, *Iterative Methods for Sparse Linear Systems*. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 2003.

[16] Y. Saad and M. H. Schultz, “Gmres: a generalized minimal residual algorithm for solving nonsymmetric linear systems,” *SIAM J. Sci. Stat. Comput.*, vol. 7, pp. 856–869, July 1986. [Online]. Available: http://portal.acm.org/citation.cfm?id=14063.14074

[17] “Intel Math Kernel Library for Linux* OS,” Document Number: 314774-005US, October 2007, Intel Corporation.

[18] Z. Bai, J. Demmel, J. Dongarra, A. Ruhe, and H. van der Vorst, *Templates for the Solution of Algebraic Eigenvalue Problems*. Philadelphia: SIAM, 2000.

[19] “Intel C++ Compiler Options,” Intel Corporation, document Number: 307776-002US.

[20] *NVIDIA CUDA CUBLAS Library Programming Guide*, 1st ed., NVIDIA Corporation, June 2007.

[21] N. Bell and M. Garland, “Efficient sparse matrix-vector multiplication on CUDA,” Dec. 2008.

recent issues

No. 2010-07 Werner Augustin, Vincent Heuveline, Jan-Philipp Weiss: Convey HC-1 Hybrid Core Computer – The Potential of FPGAs in Numerical Simulation

No. 2010-06 Hartwig Anzt, Werner Augustin, Martin Baumann, Hendryk Bockelmann, Thomas Gengenbach, Tobias Hahn, Vincent Heuveline, Eva Ketelaer, Dimitar Lukarski, Andrea Otzen, Sebastian Ritterbusch, Björn Rocker, Staffan Ronnås, Michael Schick, Chandramowli Subramanian, Jan-Philipp Weiss, Florian Wilhelm: HiFlow$^3$ – A Flexible and Hardware-Aware Parallel Finite Element Package

No. 2010-05 Martin Baumann, Vincent Heuveline: Evaluation of Different Strategies for Goal Oriented Adaptivity in CFD – Part I: The Stationary Case

No. 2010-04 Hartwig Anzt, Tobias Hahn, Vincent Heuveline, Björn Rocker: GPU Accelerated Scientific Computing: Evaluation of the NVIDIA Fermi Architecture; Elementary Kernels and Linear Solvers

No. 2010-03 Hartwig Anzt, Vincent Heuveline, Björn Rocker: Energy Efficiency of Mixed Precision Iterative Refinement Methods using Hybrid Hardware Platforms: An Evaluation of different Solver and Hardware Configurations

No. 2010-02 Hartwig Anzt, Vincent Heuveline, Björn Rocker: Mixed Precision Error Correction Methods for Linear Systems: Convergence Analysis based on Krylov Subspace Methods

No. 2010-01 Hartwig Anzt, Vincent Heuveline, Björn Rocker: An Error Correction Solver for Linear Systems: Evaluation of Mixed Precision Implementations

No. 2009-02 Rainer Buchty, Vincent Heuveline, Wolfgang Karl, Jan-Philipp Weiß: A Survey on Hardware-aware and Heterogeneous Computing on Multicore Processors and Accelerators

No. 2009-01 Vincent Heuveline, Björn Rocker, Staffan Ronnas: Numerical Simulation on the SiCortex Supercomputer Platform: a Preliminary Evaluation