Energy-efficient and Error-resilient Iterative Solvers for Approximate Computing

Schöll, Alexander; Braun, Claus; Wunderlich, Hans-Joachim

Proceedings of the 23rd IEEE International Symposium on On-Line Testing and Robust System Design (IOLTS'17) Thessaloniki, Greece, 3-5 July 2017

doi: http://dx.doi.org/10.1109/IOLTS.2017.8046244

Abstract: Iterative solvers like the Preconditioned Conjugate Gradient (PCG) method are widely-used in compute-intensive domains including science and engineering that often impose tight accuracy demands on computational results. At the same time, the error resilience of such solvers may change in the course of the iterations, which requires careful adaption of the induced approximation errors to reduce the energy demand while avoiding unacceptable results. A novel adaptive method is presented that enables iterative Preconditioned Conjugate Gradient (PCG) solvers on Approximate Computing hardware with high energy efficiency while still providing correct results. The method controls the underlying precision at runtime using a highly efficient fault tolerance technique that monitors the induced error and the quality of intermediate computational results.

Energy-efficient and Error-resilient Iterative Solvers for Approximate Computing

Alexander Schöll, Claus Braun and Hans-Joachim Wunderlich

Institute of Computer Architecture and Computer Engineering, University of Stuttgart
Pfaffenwaldring 47, D-70569, Germany, Email: {wu,braun,schoell}@informatik.uni-stuttgart.de

***Abstract*—Iterative solvers like the *Preconditioned Conjugate Gradient* (PCG) method are widely-used in compute-intensive domains including science and engineering that often impose tight accuracy demands on computational results. At the same time, the error resilience of such solvers may change in the course of the iterations, which requires careful adaption of the induced approximation errors to reduce the energy demand while avoiding unacceptable results.**

A novel adaptive method is presented that enables iterative Preconditioned Conjugate Gradient (PCG) solvers on Approximate Computing hardware with high energy efficiency while still providing correct results. The method controls the underlying precision at runtime using a highly efficient fault tolerance technique that monitors the induced error and the quality of intermediate computational results.

***Index Terms*—Approximate Computing, Energy-efficiency, Fault Tolerance, Quality Monitoring**

I. Introduction

The approximate computing paradigm allows to trade-off precision for efficiency gains and spans the whole system stack from hardware up to software and algorithms [1]. Different applications in multimedia and signal processing, for instance, are often not expected to compute *perfect* inputs and outputs and therefore exhibit an error resilience to certain numerical errors. This inherent error resilience is successfully exploited by different approximation techniques to achieve reductions in runtime, area, and energy demand. The paper at hand extends the application field of approximation techniques to the area of scientific computing.

Energy-efficient and error-resilient computing techniques are essential demands of compute-intensive domains like science and engineering. With energy demands already being a constraining factor, the approximate computing paradigm has to be applied to applications in such domains to overcome future energy challenges. At the same time, the applied approximation techniques must be carefully *controlled* and *monitored* at runtime to satisfy the tight accuracy demands that are often imposed by these domains on the computed results.

The *Preconditioned Conjugate Gradient* (PCG) method is one of the most widely used iterative solvers for large linear equations of the form $Ax = b$. Although such convergent tasks are often considered to be error-resilient [2], iterative solvers like the PCG solver can be highly sensitive to numerical errors [3]. While some applications exhibit different degrees of error resilience in different application parts, iterative solvers like the PCG solver exhibit a changing error resilience during runtime. Even single errors can significantly increase the number of iterations required for convergence to a correct result or corrupt the result without indication. This insight renders approximation techniques relying on single degrees of precision unsuitable. Instead, approximation techniques with configurable degrees of precision like [4–6] must be applied and adapted according to the changing error resilience at runtime. To ensure energy-efficient and error-resilient PCG executions with correct results, the error resilience must be continuously evaluated with low runtime and energy overhead.

Different related works like [3] begin iterative methods at the lowest available degree of precision which is increased if certain solver properties are violated. However, these properties are evaluated using expensive matrix-vector operations that may cancel the achieved energy reductions. In our previous work [7], we enabled the PCG method on approximate hardware and showed possible reductions in the hardware utilization.

In the work at hand, we present a method that enables energy-efficient and error-resilient PCG executions on approximate computing hardware. An error resilience estimation scheme monitors the accumulation of approximation errors in the *solver residual* with low runtime and energy overhead and triggers precision changes when required. At the same time, the efficient fault tolerance technique from previous work [8] ensures correct results and low iteration overheads. In experimental results, we evaluate the achieved energy-efficiency by quantifying the energy demand and show the minimum precision required for convergence to correct results.

II. State of the Art

Different related works have applied the approximate computing paradigm at different layers of the system stack [9]. The range of approximate hardware includes approximate adders [10] and multipliers [11] as well as configurable hardware [4–6] that allows to change the underlying precision at runtime. On the software level, task skipping [12] and neural networks [13] are exploited to compute approximate results.

The investigation of approximate computing schemes in the scientific and engineering computing domain is an active research area. In [14], an approximate *Cholesky decomposition* for well-conditioned problems is presented that skips insignificant values in arithmetic computations. A method is presented in [15] that enables the iterative *Lanczos algorithm* on approximate hardware by increasing the underlying precision after periodic reorthogonalization steps. The error resilience of computing the *inverse matrix p-th roots* using *Newton iterations* was investigated in [16]. The technique in [3] proposes to start the iterative solver at the lowest available degree of precision, which is increased if the underlying *optimization function* is violated. For the PCG solver, the underlying optimization function is $\min_x E(x) := \frac{1}{2}x^T Ax - x^T b$, which has to be additionally computed to detect violations. However, the introduced matrix operation can be expensive and is able to cancel out potential energy savings.

III. The Preconditioned Conjugate Gradient Solver on Approximate Computing Hardware

The PCG method solves linear equations of the form $Ax = b$ by computing improved *intermediate results* x_k in each iteration k that approach the solution x and minimize the *residual* $\delta_k := ||b - Ax_k||_2$ over time. Solver iterations are performed until the residual δ_k satisfies some result accuracy (e.g. $\delta_k < \epsilon$) which allows to accept the intermediate result x_k as a result. *Correct results* are ensured, if the inherent *convergence invariants* are satisfied for successive PCG iterations. PCG represents the solution x as a linear combination of *search directions* $\{p_0, p_1, p_2, ..., p_N\}$ and $x = x_0 + \sum_{k \le N} \alpha_k p_k$, which have to be *A-orthogonal* with

$$p_k A p_i \approx 0, k \neq i \tag{1}$$

to achieve correct convergence. At the same time, the internally used *residual variable* δ'_k must represent the actual residual with

$$\delta'_k \approx \delta_k := ||b - Ax_k||_2. \tag{2}$$

Approximation errors can violate these invariants over time which can cause additional iterations or wrong results despite apparent convergence. For this reason, the induced approximation error must be limited to avoid canceling the achieved energy reductions.

Our proposed method achieves correct results with reduced energy demands by adapting the underlying precision to the changing error resilience in PCG. The underlying idea is to periodically estimate the error resilience and to evaluate these estimations using the fault tolerance technique presented in previous work [8]. This fault tolerance technique evaluates the inherent convergence invariants efficiently to detect violations with high coverage.

Based on the rounding error investigation for PCG in [17], our proposed method estimates the error resilience with respect to the *accumulation of approximation errors*. For each available degree of precision $\varepsilon \in \{\varepsilon_0, \varepsilon_1, \varepsilon_2, ..., \varepsilon_N\}$, the method *estimates* the minimum residual δ_ε to which PCG can be resilient at runtime. This estimation is performed at runtime for each iteration k with

$$\delta_{\varepsilon,k+1} := \delta_{\varepsilon,k} + 2\alpha_k ||w_k||_2 \cdot \varepsilon \tag{3}$$

and $\delta_{\varepsilon,0} := 0$ while w_k denotes the result of $w_k := Ap_k$ that is computed in each iteration [7]. The minimum residual is found (i.e. $\delta_\varepsilon := \delta_{\varepsilon,k}$) when the residual δ'_k drops below this threshold [17] with

$$\delta'_k \le \delta_{\varepsilon,k}. \tag{4}$$

For successive iterations l with smaller residuals $\delta'_l < \delta_\varepsilon$, the precision is increased to ε' and the estimation progress is repeated. At the same time, the precision is decreased to ε, if the residual exceeds the minimum residual threshold δ_ε, which can occur due to, for instance, residual oscillations.

This estimation procedure has to be performed only once for any selected matrix A. The collected minimum residuals for the different precisions can be applied to different solver executions that are based on the same or highly similar coefficient matrices A to save one inner product and to reduce the energy demand.

IV. Experimental Results

The proposed technique is evaluated with respect to *required PCG iterations* for correct convergence and *energy reductions* in double-precision floating-point arithmetic. The approximation was applied to the most energy-demanding arithmetic operation in the dominant operation in PCG, namely the floating-point multiplications [4] in sparse matrix-vector multiplications. This approximate floating-point multiplication truncates the k least significant bits in the operand mantissa and fills the truncated part in the result mantissa with a random pattern.

To evaluate the energy demand, the operations in PCG were mapped to gate-level timing simulations that were accelerated by the GPU simulator presented in [18]. The simulations were performed for the gate-level descriptions of a double-precision floating-point adder and multiplier that were synthesized using the *NanGate* $45\,\mathrm{nm}$ library [19]. Using the energy information provided with this standard cell library, the obtained *Weighted Switching Activity* results were translated to power and energy results. While the multiplier contains 20,812 two-input gates and a critical path delay of $14.21\,\mathrm{ns}$, the adder contains 5,678 gates and a critical path delay of $14.98\,\mathrm{ns}$. The static power dissipation is $0.154\,\mathrm{mW}$ and $0.033\,\mathrm{mW}$, respectively. In this work, the term *precise hardware* refers to an IEEE 754-compliant floating-point unit. As benchmarks, 26 matrices from the Florida Sparse Matrix Collection [20] were evaluated that comprise 15,844 to 10,614,210 non-zero elements. More than 500,000 experiments were performed to derive the experimental results discussed below. All evaluated experiments converged to a correct solution.

Figure 1 shows the number of iterations and the demanded energy compared to the execution on precise hardware. The evaluated matrices are ordered by the number of non-zero elements. The increase in the number of iterations ranges from 0.0% to 25.4% and is on average only 4.9%. For 18 matrices, the number of iterations is only increased by at most 4%. At the same time, an energy reduction can be observed for 22 matrices compared to executing PCG on precise hardware.

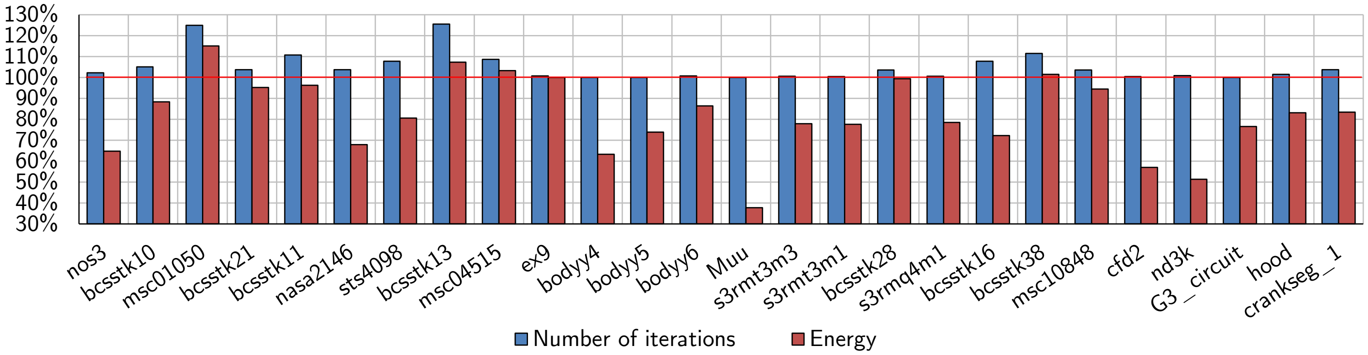

Fig. 1. Number of iterations and energy demand of PCG on approximate hardware compared to the execution on precise hardware.

For these matrices, the energy demand is on average reduced by 22.5% and in total up to 62.3%. For 5 matrices, the energy demand is reduced by at least 35%. These reductions of hardware utilization can be explained by the efficiency of the underlying fault tolerance technique that induces only low runtime overhead to evaluate the inherent solver properties.

V. Conclusion

In this work, we presented a method that enables the Preconditioned Conjugate Gradient (PCG) solver on approximate computing hardware and ensures energy-efficient and error resilient solver executions. Although such convergent tasks are typically considered to be inherently error-resilient, numerical approximation errors can cause additional iterations or wrong results. Our proposed method controls the underlying precision along the changing error resilience of the solver to ensure correct results while reducing the error demand. This method estimates the underlying error resilience by monitoring the accumulation of approximation errors in the underlying variables. An efficient fault tolerance technique evaluates these estimations to detect violations of the inherent solver invariants. Experimental results showed energy reductions in 22 out of 26 evaluated benchmark matrices. For these matrices, the energy demand was reduced by up to 62.3% while the iteration overhead is on average 4.9%.

VI. Acknowledgment

The authors would like to thank the German Research Foundation (DFG) for financial support of the project within the Cluster of Excellence in Simulation Technology (EXC 310/2) at the University of Stuttgart.

Bibliography

[1] J. Han and M. Orshansky, "Approximate computing: An emerging paradigm for energy-efficient design", in *Proc. 18th IEEE European Test Symposium (ETS'13)*, May 2013, pp. 1–6.

[2] V. Chippa, S. Chakradhar, K. Roy, and A. Raghunathan, "Analysis and Characterization of Inherent Application Resilience for Approximate Computing", in *Proc. 50th ACM/EDAC/IEEE Design Automation Conference (DAC'13)*, May 2013, pp. 1–9.

[3] Q. Zhang, F. Yuan, R. Ye, and Q. Xu, "ApproxIt: An Approximate Computing Framework for Iterative Methods", in *Proc. 51st ACM/EDAC/IEEE Design Automation Conference (DAC'14)*, 2014, pp. 53.1:1–6.

[4] H. Zhang, W. Zhang, and J. Lach, "A Low-Power Accuracy-Configurable Floating Point Multiplier", in *IEEE International Conference on Computer Design (ICCD)*, Seoul, South Korea, Oct. 2014, pp. 48–54.

[5] C. Liu, J. Han, and F. Lombardi, "A Low-Power, High-Performance Approximate Multiplier with Configurable Partial Error Recovery", in *Proc. of the Conference on Design, Automation & Test in Europe (DATE'14)*, 2014, pp. 95:1–95:4.

[6] V. Camus, J. Schlachter, and C. Enz, "Energy-Efficient Inexact Speculative Adder with High Performance and Accuracy Control", in *IEEE Intl. Symp. on Circuits and Systems (ISCAS)*, 2015, pp. 45–48.

[7] A. Schöll, C. Braun, and H.-J. Wunderlich, "Applying Efficient Fault Tolerance to Enable the Preconditioned Conjugate Gradient Solver on Approximate Computing Hardware", in *Proc. IEEE Intl. Symp. on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT'16)*, Sep. 2016, pp. 21–26.

[8] A. Schöll, C. Braun, M. A. Kochte, and H.-J. Wunderlich, "Efficient Algorithm-Based Fault Tolerance for Sparse Matrix Operations", in *to appear in Proceedings of The 46th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN'16)*, Toulouse, France, Jun. 2016.

[9] S. Mittal, "A Survey Of Techniques for Approximate Computing", *ACM Computing Surveys*, vol. 48, no. 4, p. 62, 2016.

[10] J. Hu and W. Qian, "A New Approximate Adder with Low Relative Error and Correct Sign Calculation", in *Proc. of the Design, Automation & Test in Europe Conference & Exhibition*, 2015, pp. 1449–1454.

[11] L. Qian, C. Wang, W. Liu, F. Lombardi, and J. Han, "Design and Evaluation of an Approximate Wallace-Booth Multiplier", in *IEEE Intl. Symp. on Circuits and Systems (ISCAS)*, 2016, pp. 1974–1977.

[12] S. Sidiroglou-Douskos, S. Misailovic, H. Hoffmann, and M. Rinard, "Managing Performance vs. Accuracy Trade-offs with Loop Perforation", in *Proc. of the 13th Europ. Conf. on Foundations of Software Engineering*, 2011, pp. 124–134.

[13] T. Moreau, M. Wyse, J. Nelson, A. Sampson, H. Esmaeilzadeh, L. Ceze, and M. Oskin, "SNNAP: Approximate Computing on Programmable SoCs via Neural Acceleration", in *IEEE Intl. Symp. on High Performance Computer Architecture (HPCA)*, 2015, pp. 603–614.

[14] M. Schaffner, F. K. Gürkaynak, A. Smolic, H. Kaeslin, and L. Benini, "An Approximate Computing Technique for Reducing the Complexity of a Direct-solver for Sparse Linear Systems in Real-time Video Processing", in *Proc. 51st ACM/EDAC/IEEE Annual Design Automation Conference (DAC'14)*, San Francisco, CA, Jun. 2014, pp. 66:1–6.

[15] Q. Zhang, Y. Tian, T. Wang, F. Yuan, and Q. Xu, "ApproxEigen: An Approximate Computing Technique for Large-Scale Eigen-Decomposition", in *Proc. of the IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2015, pp. 824–830.

[16] M. Lass, T. D. Kühne, and C. Plessl, "Using approximate computing for the calculation of inverse matrix p-th roots", 2017.

[17] S. Cools, W. Vanroose, E. F. Yetkin, E. Agullo, and L. Giraud, "On rounding error resilience, maximal attainable accuracy and parallel performance of the pipelined Conjugate Gradients method for large-scale linear systems in PETSc", in *Proc. of the Exascale Applications and Software Conference*. ACM, 2016, p. 3.

[18] S. Holst, M. E. Imhof, and H.-J. Wunderlich, "High-Throughput Logic Timing Simulation on GPGPUs", *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, vol. 20, no. 3, pp. 1–22, 2015.

[19] Nangate Inc., "45nm open cell library", *http://www.nangate.com*.

[20] T. A. Davis and Y. Hu, "The University of Florida Sparse Matrix Collection", *ACM Trans. on Mathematical Software*, vol. 38, no. 1, pp. 1:1–1:25, Nov. 2011.