Algorithm-Based Fault Tolerance for Many-Core Architectures

Claus Braun, Hans-Joachim Wunderlich
Institute of Computer Architecture and Computer Engineering
University of Stuttgart
Pfaffenwaldring 47, D-70569 Stuttgart, Germany
email: {braun, wu}@informatik.uni-stuttgart.de

***Abstract*—Modern many-core architectures with hundreds of cores provide a high computational potential. This makes them particularly interesting for scientific high-performance computing and simulation technology. Like all nano scaled semiconductor devices, many-core processors are prone to reliability harming factors like variations and soft errors. One way to improve the reliability of such systems is software-based hardware fault tolerance. Here, the software is able to detect and correct errors introduced by the hardware. In this work, we propose a software-based approach to improve the reliability of matrix operations on many-core processors. These operations are key components in many scientific applications.**

I. Introduction

In this work, known techniques for algorithm-based fault tolerance are analyzed and adapted to emerging many-core architectures in order to tackle soft errors that manifest themselves at software level through erroneous values. The proposed approach does not rely on any modification of the hardware. We focus on matrix operations, particularly on the matrix multiplication. The approach has been implemented and evaluated on a Nvidia Tesla GPGPU system [1], nevertheless it is general enough to be applied to other many-core platforms in a straight forward manner. As software-backend, the Nvidia CUDA environment is used.

II. Algorithm-Based Fault Tolerance (ABFT)

ABFT [2] is a software technique to improve reliability at system level. Algorithms are modified to work on encoded data and to produce encoded data. The encoding introduces redundant information that enables the detection and correction of errors. In case of the matrix multiplication, the encoding can be done by column and row checksums which are stored within the matrices.

The used many-core hardware platform follows an execution paradigm which we call many-threading. Thousands of threads execute the same code in parallel. This allows a very high throughput, especially for data-parallel problems. For an efficient mapping and integration of ABFT on such a platform, three major aspects have to be considered together: Data, ABFT and many-threading. Our approach unifies all of these aspects by using a block-based mapping. Here, the data (matrices) is partitioned into blocks (sub-matrices) and all operations are parallelized over these blocks. To achieve optimum protection of the data, the ABFT scheme and the encoding are also applied at the level of these blocks. Finally, the many-threading is incorporated this way, because the threads are grouped into blocks for execution.

The block-based approach provides several advantages. From the hardware perspective, the partitioning of the data allows the usage of small local memories on the GPGPU as caches. This improves the overall performance significantly. For the ABFT encoding, it was shown in [3], that smaller encoder vectors improve the numerical properties of the checksums. With the approach presented here, the encoding is done at the level of small sub-matrices, instead of the whole matrix. Another important advantage is the vastly improving error detection and correction capability. The original ABFT scheme could only detect and correct a very limited number of errors. With the block-based approach, the number of detectable and correctable errors increases with the number of blocks. Therefore, a high degree of fault tolerance can be guaranteed, even for very large matrices.

III. Results

The results show that ABFT schemes can be incorporated into matrix operations on many-core architectures with low performance overhead. The diagrams depict the overhead and performance comparison between an unprotected multiplication, the ABFT scheme and the Duplication with Comparison approach. The achievable error detection rates depend on the

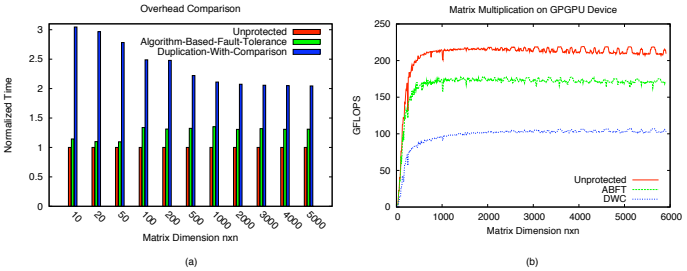

Fig. 1. Overhead and Performance Comparison

input data, therefore we allow the user to define a threshold value that fits his needs and does not cause too many false positives.

References

[1] Nvidia, http://www.nvidia.com/tesla.
[2] K.-H. Huang and J. Abraham, "Algorithm-based fault tolerance for matrix operations," *IEEE Transactions on Computers*, vol. 33, no. 6, pp. 518–528, 1984.
[3] J. Rexford and N. Jha, "Algorithm-based fault tolerance for floating-point operations in massively parallel systems," in *Circuits and Systems, 1992. ISCAS '92. Proceedings., 1992 IEEE International Symposium on*, vol. 2, May 1992, pp. 649–652 vol.2.