Parallelization & Checkpointing of GPU Applications Through Program Transformation

Lizandro Damian Solano-Quinde

Abstract

GPUs have emerged as a powerful tool for accelerating general-purpose applications. The availability of programming languages that makes writing general-purpose applications for running on GPUs tractable have consolidated GPUs as an alternative for accelerating general-purpose applications. Among the areas that have benefited from GPU acceleration are: signal and image processing, computational fluid dynamics, quantum chemistry, and, in general, the High Performance Computing (HPC) Industry.

In order to continue to exploit higher levels of parallelism with GPUs, multi-GPU systems are gaining popularity. In this context, single-GPU applications are parallelized for running in multi-GPU systems. Furthermore, multi-GPU systems help to solve the GPU memory limitation for applications with large application memory footprint. Parallelizing single-GPU applications has been approached by libraries that distribute the workload at runtime, however, they impose execution overhead and are not portable. On the other hand, on traditional CPU systems, parallelization has been approached through application transformation at pre-compile time, which enhances the application to distribute the workload at application level and does not have the issues of library-based approaches. Hence, a parallelization scheme for GPU systems based on application transformation is needed.

Like any computing engine of today, reliability is also a concern in GPUs. GPUs are vulnerable to transient and permanent failures. Current checkpoint/restart techniques are not suitable for systems with GPUs. Checkpointing for GPU systems present new and interesting challenges, primarily due to the natural differences imposed by the hardware design, the memory subsystem architecture, the massive number of threads, and the limited amount of synchronization among threads. Therefore, a checkpoint/restart technique suitable for GPU systems is needed.

The goal of this work is to exploit higher levels of parallelism and to develop support for application-level fault tolerance in applications using multiple GPUs. Our techniques reduce the burden of enhancing single-GPU applications to support these features. To achieve our goal, this work designs and implements a framework for enhancing a single-GPU OpenCL application through application transformation.