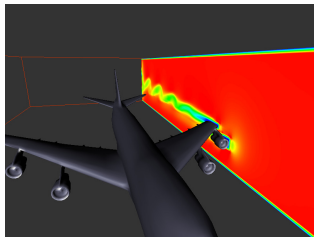
Towards Dynamic Reconfigurable Load-Balancing for Hybrid Desktop Platforms

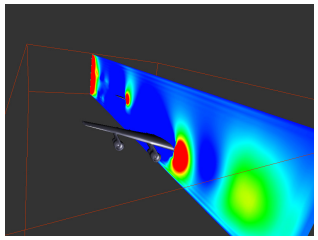
Alécio P. D. Binotto (abinotto@inf.ufrgs.br) – Carlos E. Pereira – Dieter W. Fellner

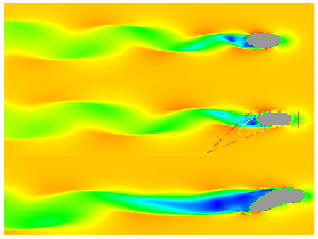
Motivation

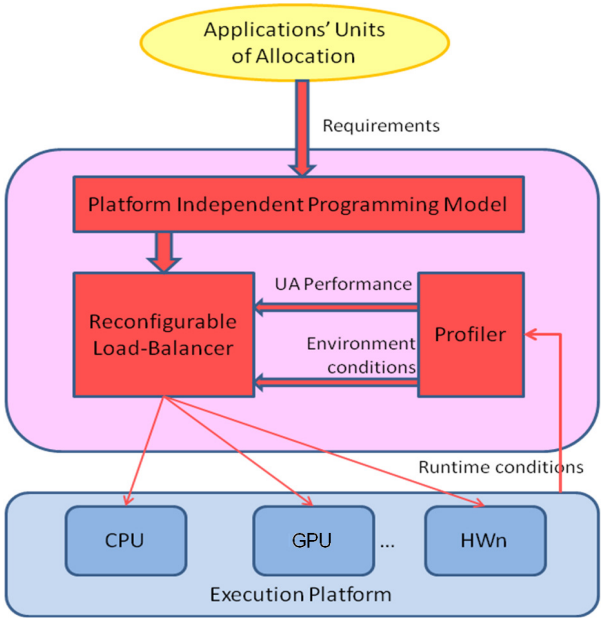
Desktop accelerators (like GPUs) form a powerful heterogeneous platform in conjunction with multi-core CPUs. To improve application performance on these heterogeneous platforms, load-balancing plays an important role to distribute workload. And even more important is to consider online parameters that cannot be known a priori for system scheduling.

A Computational Fluid Dynamics (CFD) application, developed at the Fraunhofer IGD, is used to exemplify the implementation of iterative SLE (Systems of Linear Equations) solvers and the need of a hybrid CPU-GPU approach to optimize performance.









Innovation

The approach abstracts the Pus using the OpenCL API as the platform independent programming model. It has the proposal to extend OpenCL with a module that schedule and balance the workload over the CPU and GPU for the specific case study in a high level. Starting with an initial scheduling configuration just when the application starts, an online profiler monitors and stores tasks' execution times and platform conditions. Since the tasks are non-deterministic, during application execution, a reconfigurable dynamic scheduling is performed considering changes on runtime conditions. Figure above depicts the approach; and bellow, the tow-phase scheduling approach.

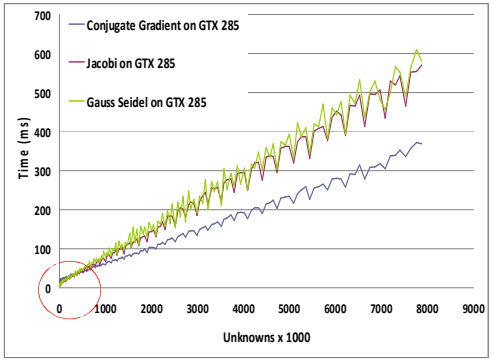
First Assignment: the first guess faces a multidimensional scheduling problem with NP-hard complexity and become more complex when dealing with more than two PUs and several tasks. To optimize, this assignment is based on heuristics taking into account the performance benchmark described on the right. The first scheduling is performed in a static-code way, but dynamically just after the application starts considering the domain size and the premise that the PUs are idle, defining a rule based on the break-even point values presented on the next section. However, this strategy can lead to a PU overflow, i.e., tasks being scheduled to the same PU, showing the need for a **context-aware** adaptation.

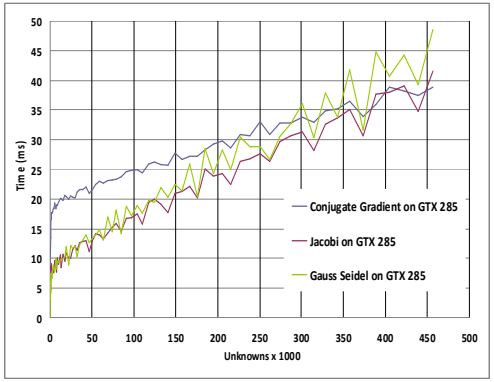
Dynamic Reconfiguration: After the first assignment, information provided by profiling is considered. Based on estimated costs, dynamic parameters, and awareness of runtime conditions, a new task can be reconfigured to other PU just if the estimated time to be executed in the new PU is less than the time in the current PU.

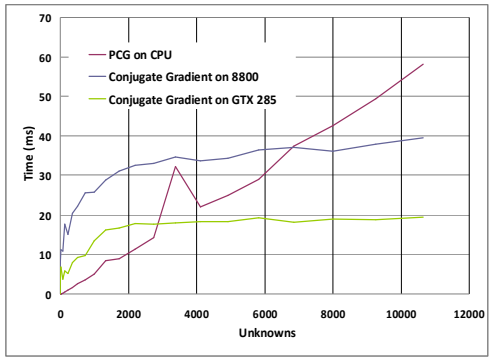
Preliminary Results

Specially, 3 iterative methods for solving SLEs are analyzed: Jacobi, red-black Gauss-Seidel (GS), Conjugate Gradient (CG). The break-even points over the PUs (quad-core CPU, 8800GT GPU, and GTX285 GPU) used in the first assignment of the scheduling are presented as first results.

The figures bellow show the performance of the solvers on the most powerful used PU, the GTX285 GPU. For a small problem size, GS and Jacobi are faster; and for large problems, it is the CG. On the right, a comparison of the CG over the PUs. For problems till 2K unknowns, the CPU has better performance than GPUs.





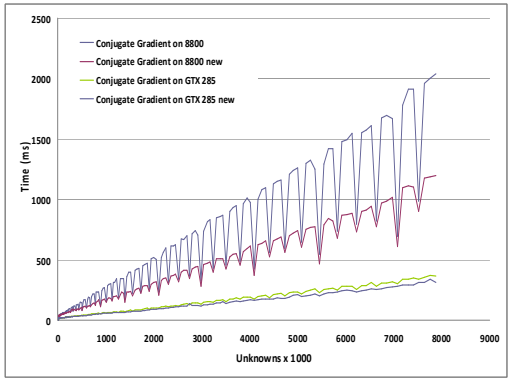


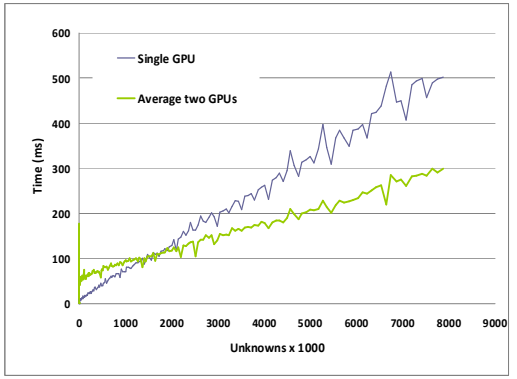
The GPU implementation was made using CUDA and optimized for memory coalescing. Particularly, the figure on the right side, above, compares the CG without and with (new) our strategy for enabling memory coalescing.

The CG was also used for comparing performances using multiple GPUs. There is a need of 2M unknowns for using two GPUs. With less elements, it results on an increasing of communication. The multi-GPU approach demonstrates that the speedup depends on the problem size and the achieved speedup was 1.7 for 8M unknowns.

Importance

This PhD research follows the state-of-the-art as several scientific applications can now be executed on multi-core desktop platforms. To our knowledge, there is a need for research oriented to support load-balancing over a CPU-GPU (or CPU-accelerators) platform. The work shows its relevance analyzing not just platform characteristics, but also the platform context execution scenario.





Conclusions and Remaining Challenges

Based on the performance evaluation, it was verified the need of strategies for dynamic scheduling, improving OpenCL. We propose a method for dynamic load-balancing over CPU and GPU, applied to solvers for SLEs. As remaining challenges, to develop the load-balance reconfiguration phase on the presented case study; and to analyze the performance of the application without the proposed method to evaluate the strategy overhead.

Acknowledgments
Thanks for Daniel Weber and Christian Daniel for their support. A. Binotto thanks the support given by DAAD and Alßan, scholarship no. E07D402961BR.