Strategies to Hide Communication for a Classical Molecular Dynamics Proxy Application

Issakar Ngatang and Masha Sosonkina

Summary
Co-designing applications and computer architectures has become of major importance due to the growing complexity of both applications and architectures and the need to better match application characteristics to the available hardware. Thus, "mini-applications", which serve as proxies of large-scale ones by highlighting their most intensive parts and major workflow components, have appeared to the co-design, tuning, and adaptation purposes. This paper presents a work on optimizing the communication subsystem of a classical MD proxy (CoMD) application executed on multi-core computing clusters. The research focuses on hiding communication with certain buffer handling operations. In particular, two strategies are presented: one that uses two parallel threads for communication and buffer handling and another that introduces more parallelism by allowing all the available threads to unload the buffers while using two thread to communicate, thereby improving load balancing. The first proposed strategy yields performance gains up to 61% in the communication routines, corresponding to 6% gains in the overall time, while the second strategy achieves, respectively, about 73% and 6.3% improvement.

Shared-Memory Parallelization of the Semi-Ordered Fast Iterative Method

Josef Weinbub, Florian Dang, Tor Gillberg and Siegfried Selberherr

Summary
The semi-ordered fast iterative method allows to compute monotone front propagation of anisotropic nature by solving the eikonal equation. Compared to established iterative methods, such as the fast iterative method, the semi-ordered fast iterative method offers increased stability for variations in the front velocity. So far, the method has only been investigated in a serial, two-dimensional context; therefore, we investigate in this work a parallelization approach via OpenMP and evaluate it for three-dimensional problems, being especially of interest to real-world applications. We discuss the parallel algorithm and compare the performance as well as the computed solutions with an OpenMP-powered fast iterative method. We use different speed functions as well as varying problem sizes to investigate the impact of the computational load. We show that although the semi-ordered fast iterative method is inferior to the fast iterative method with respect to parallel efficiency, execution performance is significantly faster.

PerDome: A Performance Model for Heterogeneous Computing Systems

Li Tang, Xiaobo Sharon Hu and Richard F. Barrett

Summary
Heterogeneous systems, consisting of different types of processors, have the potential to offer higher performance at lower energy cost than homogeneous systems. However, it is rather challenging to actually achieve the high execution efficiency promised by such a system due to the larger design space and the lack of reliable performance/energy models for aiding design space exploration. This paper fills this gap by proposing a performance model for heterogeneous systems. In processor level, the roofline model can produce the performance upper bound of executed code using its ratio of computation to memory traffic. Our model, referred to as PerDome, builds on the roofline model and can reliably predict the system performance for both homogeneous execution (where each processor either executes the entire application code or none) and heterogeneous execution (where each processor executes part of the application code). Two case studies are carried out to demonstrate the effectiveness of PerDome. The results show that PerDome can indeed provide a good estimate for performance comparisons which can then be used for heterogeneous system design space exploration.

An Improved Probability-One Homotopy Map for Tracking Constrained Clustering Solutions

David Easterling, Layne Watson and Naren Ramakrishnan

Summary
This paper proposes a new homotopy map for use with constrained clustering problems, improving over previously introduced maps through the introduction of a K-Means approximation and the use of the Kreisselmeier-Steinhauser function to provide more efficient computation, as well as demonstrating through experimentation the power of this new map.

Productive Parallel Programming with Charm++

Phil Miller

Summary
Charm++ is a general purpose parallel programming framework for high-performance computing applications. It provides high productivity for developers through separation of concerns. Application programmers are responsible for providing scalable parallel algorithms that solve their problem of interest. The Charm++ runtime system is responsible for ensuring that the implementation of those algorithms executes efficiently across a wide range of platforms, and in the face of dynamic variation in work load and computational resources. This level of capability has allowed several Charm++ applications to run at full-machine scale on some of the largest supercomputers in the world.