Opportunistic Computing: A New Paradigm for Scalable Realism on Many-Cores

Romain Cledat 1  Tushar Kumar 1
Georgia Institute of Technology
romain@gatech.edu, tushark@ece.gatech.edu

1. Motivation
For a growing number of applications, expressed realism and not performance in FLOPS is what matters. For example, in a live video encoding application, realism translates to how well the transmitted compressed video reflects the original source. On single-cores, programming techniques to maximize FLOPS and realism exist. For multi-cores, maximizing FLOPS is difficult as applications must harness parallel resources. However, maximizing realism is a more achievable goal. This work focuses on developing techniques to achieve this goal. Realism by itself is also not a sufficient design goal. Indeed, a programmer typically tries to balance the amount of realism obtainable with the time it takes to obtain it. Furthermore, for certain applications with a high level of user-interactivity such as games, the responsiveness of an application is also an important design consideration. Therefore, the design goal for modern applications is to maximize realism under resource and responsiveness constraints.

2. Our approach for multi-cores
Traditional parallelization techniques, such as task and data parallelism, are often difficult to apply in practice given the complexity of finding and expressing parallel tasks. Furthermore, they do not address speeding up sequential parts of an application which will, according to Amdahl’s law, become the bottleneck. For example, an application whose parallelizable components constitute 60% of its execution time only has a maximum speedup potential of 2.5. We propose techniques to 1) speedup certain types of sequential algorithms and 2) maximize realism under responsiveness constraints for scalable algorithms. Both techniques make use of multi-cores in a novel way to speedup computation that was hitherto considered sequential.

Exploiting randomness For many sequential kernels, a randomized version exists. Concurrently invoking multiple independent instances of a randomized kernel considerably enhances the probability that one instance will complete with an execution time lower than the average time thus providing speedup to a sequential kernel with high probability.

In other applications where multiple heuristics can be used to perform the same computation, launching them in parallel and picking the fastest also provides speedup.

Scalable semantics In immersive applications such as gaming and multimedia, scalable semantics are present. Due to their nature, the complexity and resource needs of their algorithms can be significantly affected by certain key parameters such as the granularity of the time-step in a simulation, or the type of physical effects considered. Some algorithms can also be scaled in terms of the number of cores they can use. We propose to dynamically adjust these parameters to maximize the level of complexity, which can lead to greater realism, while constrained by resource and responsiveness constraints. Our system will adjust the computation to fill up any available slack: either unused core or unused time.

Using control theory and machine learning techniques we exploit the scalable nature of these algorithms to tune their expressed realism to currently available resources.

Framework Our framework consists of an API and a runtime that allows both models to be expressed. Specifically, our API allows the programmer to identify sequential components that can be sped up, and scalable semantics components. Our runtime will then optimally speed up the sequential components to provide more time resources to the scalable components and thus improve realism in the application.

3. Results
We show considerable speedups with the first technique when applied to a SAT solver, a randomized Hamiltonian cycle finder and a path finding algorithm. In some cases, the speedups are superlinear. For the second technique, we show that in the Torque Game Engine and the MPEG2 encoder, we can maintain a desired frame-time with very high probability by dynamically scaling the algorithms.

References