Nested Parallelism and Hierarchical Locality

Dealing with locality has proved to be one of the largest challenges in developing clean and scalable parallel codes, especially for unstructured computations. This is complicated by the fact that today's machines have multiple levels of hierarchy that need to be considered: local per-core caches, shared chip caches, memory partitioned by boards, and networks for which bandwidth varies based on distance. These levels are likely to become ever more complicated and important at the exascale. To have programmers worry about each of these levels would be near impossible and likely not portable.

The question is whether it is possible to have a high level and highly scalable model that allows one to program for and analyze hierarchical locality without having to understand details of a machine.
I will suggest an approach based on fine-grained dynamic nested parallelism. Similarly to the sequential cache-oblivious model, locality is analyzed in terms of just two "cache" parameters that cannot be used by the program. However the model accounts for parallelism by allowing arbitrary nested forking and parallel loops, along with a collection of parallel primitives. Appropriate compilers and runtime scheduler can then be used for mapping the locality and parallelism available and analyzed in the code to a variety of parallel memory hierarchies.