

Hierarchical Locale Models in Chapel

Sung-Eun Choi, David Iten, Elliot Ronaghan, Greg Titus (presenting)

Chapel Team

Cray Inc.

As HPC approaches exascale computing, processor and node architectures are gaining in both breadth and hierarchical depth, becoming heterogeneous, and diverging rapidly both between vendors and even across generations from a given vendor. But while it seems unavoidable that exascale system architectures will be complicated it also seems clear that, for the sake of programmability and portability, this complexity must be hidden from the application level.

Chapel's solution to this problem lies in *hierarchical locales*. Hierarchical locales allow user code to express the architectural model to the compiler and allow mapping high-level parallel algorithms using standard Chapel concepts down to target systems. By doing so, they shield not only the applications but also the compiler from node architecture complexity.

Hierarchical locales are implemented in terms of *locale model* classes written in Chapel itself. A locale model describes one level of the hierarchical system architecture. It identifies its place in the hierarchy and supplies a standardized functional interface to control execution and data placement. The code for Chapel constructs which exercise locality-oriented features—such as allocating variables or placing tasks near the data they operate on—is emitted in terms of this standardized interface. Chapel's higher-level abstractions, notably distributions and domain maps, are also written in Chapel itself and use these locality-oriented features extensively. Application code can be expressed algorithmically in terms of the higher-level abstractions, and without understanding the architecture itself, the compiler can use the locale models to map those higher-level abstractions to the architecture effectively.

In this talk we introduce the notion of hierarchical locales, describe the locale model class and the two existing implementations for flat and NUMA compute node architectures, and then discuss current and planned work to improve our NUMA support and add a locale model for the Intel[®] Xeon Phi[™] Knight's Landing (KNL) processor architecture.