Automatic Kernel Mapping for Functionally Heterogeneous Parallel Architectures

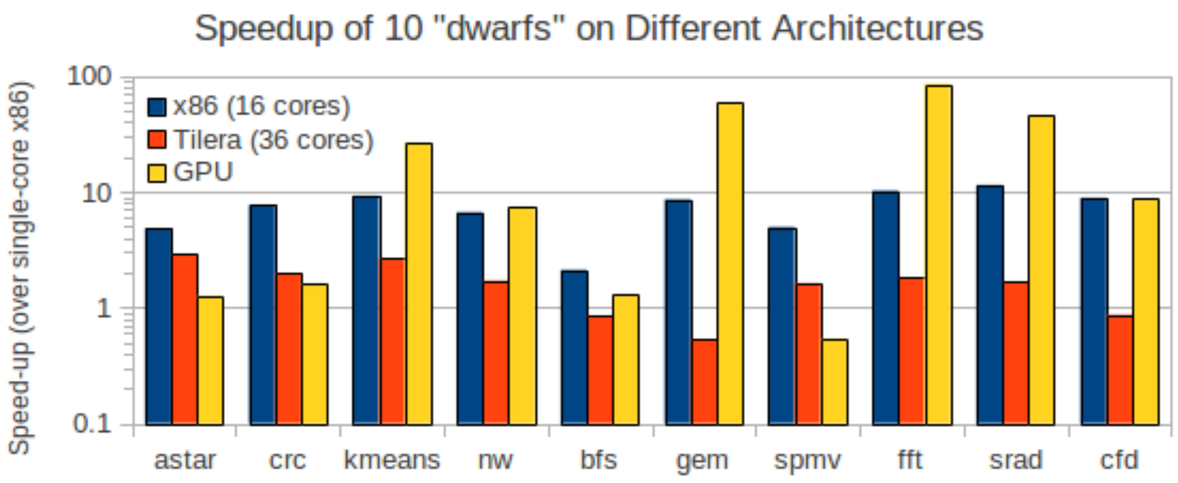
Robert Lyerly, Alastair Murray, Binoy Ravindran
Bradley Department of Electrical and Computer Engineering, Virginia Tech, Blacksburg, VA 24060
{rlyerly, alastair, binoy@vt.edu}

Systems Software Research Group
ssrg.ece.vt.edu

VirginiaTech
Invent the Future®

Motivation

- Emerging parallel architectures/accelerators can dramatically improve application performance but are difficult to program.
- Different "computational kernels" map most effectively to different architectures based on their computational features.
- New accelerators requires learning new languages/paradigms and understanding architectural characteristics.
- Many different accelerators means many different paradigms.
- Automatic tools are thus essential for helping the programmer exploit heterogeneous systems to extract maximal benefit.

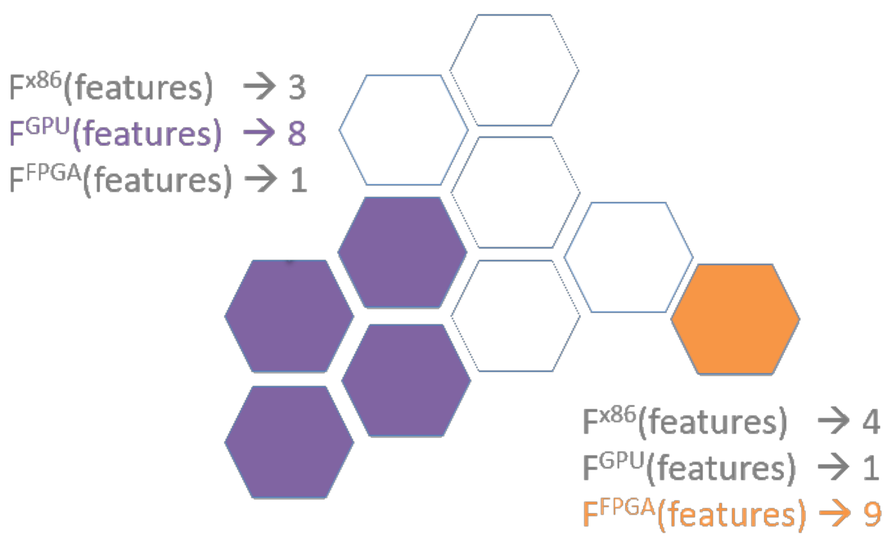


Goals

- Take program written in OpenMP (traditional multi-core parallelism) and automatically refactor to take advantage of multiple accelerators, such as GPUs.
- Use machine learning to estimate the performance of a kernel on various architectures, based on the kernel's features.
- Partition kernels across available accelerators using multiple strategies (performance, power, etc.).
- Automatically refactor the input application to handle data movement and kernel launches on the accelerators.
- Apply architecture-specific optimizations to kernels.

Analysis Model

- Partition at the call-graph level – produce one call-graph per accelerator, with accelerator-specific information annotated.
- Computational kernels are defined as sub-graphs of the call-graph with exactly one entry point.
- Node weights correspond to expected runtime, edge weights correspond to expected data transfer time per accelerator.



Design

Machine Learning

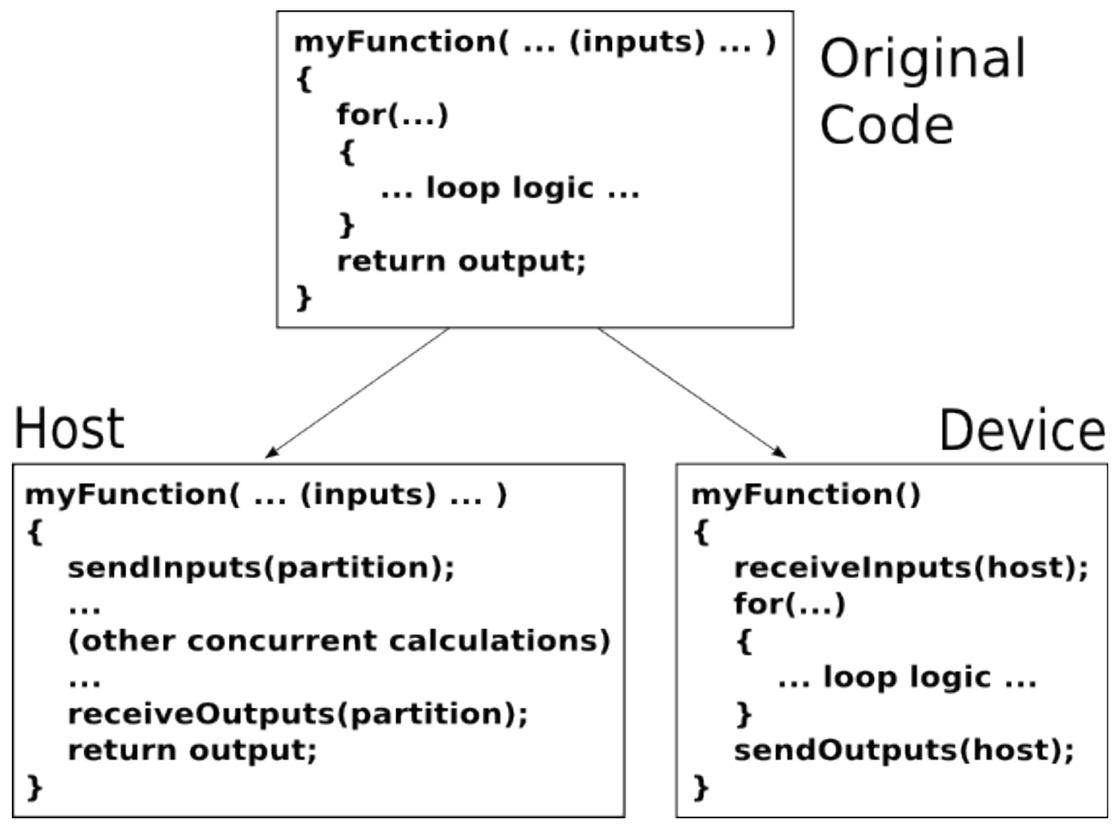
- Formally characterize each kernel by counting "features" (e.g. instruction count, integer/floating-point operations, memory access patterns, control flow complexity, etc.) per function by using a GCC plugin.
- Train offline using "Weka" machine learning tool by correlating features with runtimes on different accelerators to predict runtimes or classify kernels as suitable for different accelerators.

Partitioning Strategies

- Aim for best performance.
- Aim for lowest power (Tilera – 50W, x86 – 115W, GPU – 225W).
- Consider external workload (aim for best throughput).

Code Refactoring

- Move kernels to accelerator(s) using architecture-specific languages and frameworks, done using to tool built on top of the ROSE source-to-source compiler infrastructure and related tools.
- Automatically handle all kernel data movement (function parameters, return values, side-effects, global variables) and execution, e.g.:
  - Tilera – MPI_Send/MPI_Receive
  - GPU – cudaMalloc/cudaMemcpy/cudaFree



Conclusions

- Proposed a framework to automatically characterize and partition computational kernels across accelerators in a heterogeneous system.
- Developer writes application in OpenMP, but does not need to know the architectural details of the accelerators.
- Use machine learning to make partitioning and mapping decisions, then automatically refactor/optimize the application.

Acknowledgements
The work is supported by the U.S. Office of Naval Research under Contract N00014-12-1-0880

For more information, visit
www.popcornlinux.org