A Programming System for Future Proofing Performance Critical Libraries

Li-Wen Chang[1], Izzat El Hajj[1], Hee-Seok Kim[1], Juan Gómez-Luna[2], Abdul Dakkak[1] ,Wen-Mei Hwu[1]

[1] University of Illinois at Urbana-Champaign, Urbana, USA

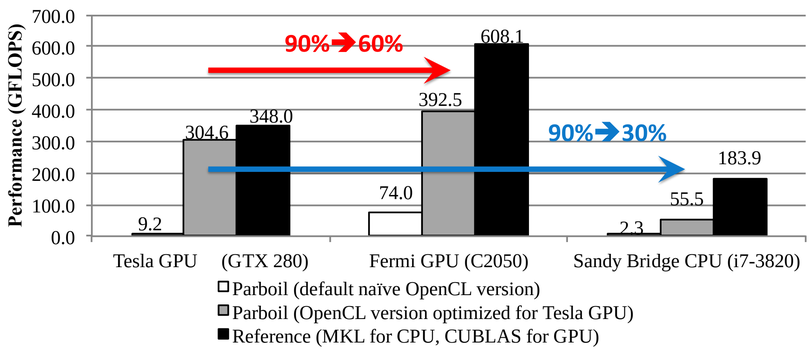
[2] Universidad de Córdoba, Córdoba, Spain

Motivation

- Maintaining optimized programs for different devices is costly
- Programs written once should run difference devices with performance, which is known performance portability

Performance Portability Issues

- Not all optimization are transferable
- OpenCL guarantees portability in functionality not performance
- A single version of OpenCL source code may not be enough



Challenges and Solutions

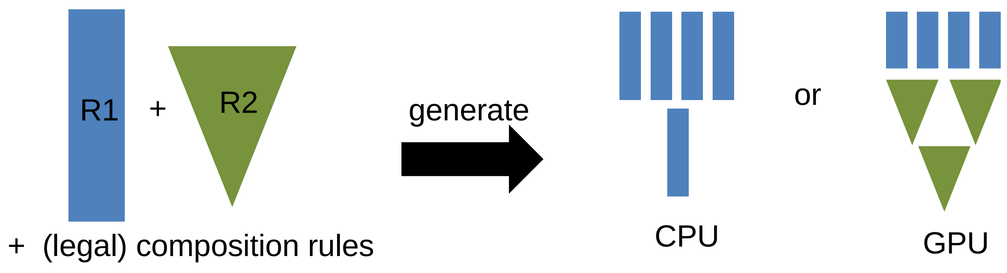
- Performance portability is a challenge because of architectural difference between various types and/or generations of devices

Differences	Granularity of Parallelism	Memory Model	Levels of Hierarchy	Resource Size	Special Instructions
Specific Solutions	Overdecomposition and coarsening	Auto data-placement Locality-aware scheduling	Nested parallelism	Autotuning	Language abstractions Pattern replacement
General Solution	Basic algorithm libraries Algorithm selection				

- We propose **TANGRAM** programming system to deliver performance portability across devices

TANGRAM Language Design

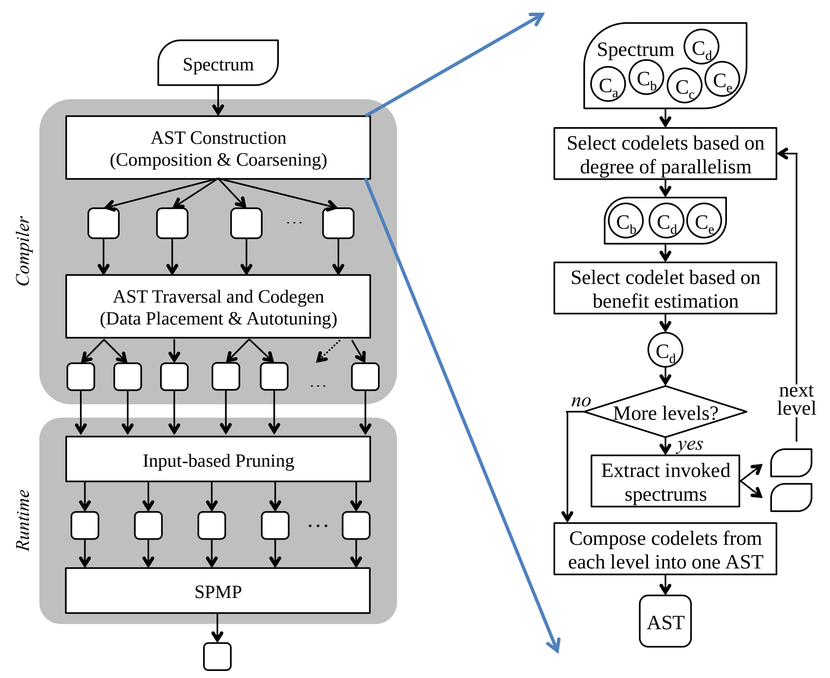
- TANGRAM adopts codelet programming model
  - A codelet is defined as a code snippet reusable for one or many kernels
- Users write interchangeable alternative codelets, and corresponding composition and partition rules for a computation pattern (called spectrum)
  - We do **Not** ask users to write multiple versions of kernels



- TANGRAM supports recursive composition to adapt different hierarchies of devices and vector codelets for SIMD architectures
- TANGRAM also provides performance tuning annotation to enable parameterization

TANGRAM Compiler Design

- TANGRAM matches AST with the hierarchies of the target device and performs code generation for the device
  - Optimizations such as data placement and fusion are built-in
- TANGRAM may generate multiple (<10) versions for runtime selection

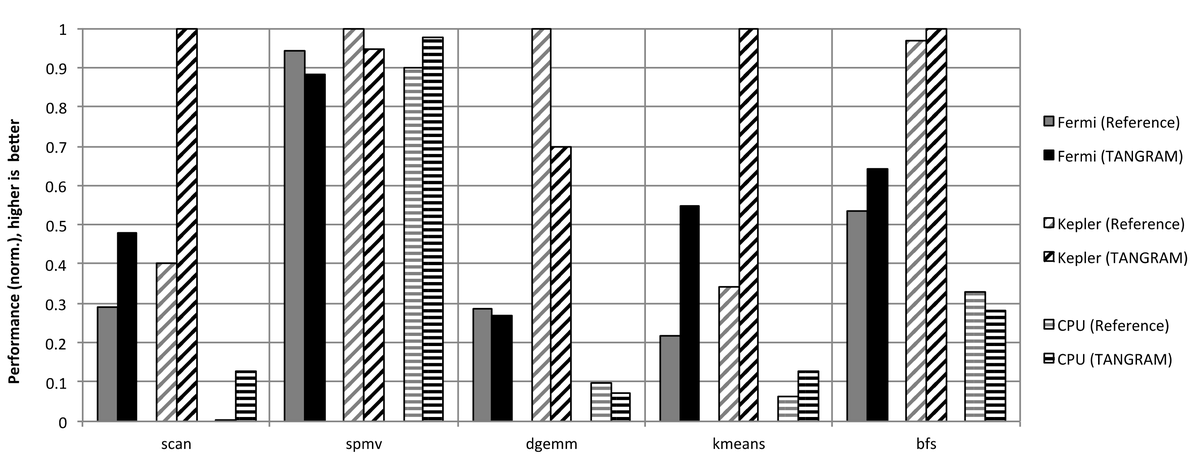


TANGRAM Runtime Design

- TANGRAM supports dynamic selection for the optimal version using a lightweight profiling technique (SPMP)
  - More details in our DySel paper, ASPLOS 2016
- TANGRAM also supports traditional static offline profiling for regular application

Experimental Results

- TANGRAM can deliver **70%** or higher performance compared to existing well-optimized libraries, such as Intel MKL, NVIDIA CUBLAS, CUSPARSE, or Thrust, or experts' optimized benchmarks, Rodinia, on different devices



Conclusion

- We propose TANGRAM, a programming system for performance portability across devices
- Our results show TANGRAM can achieve promising performance across devices