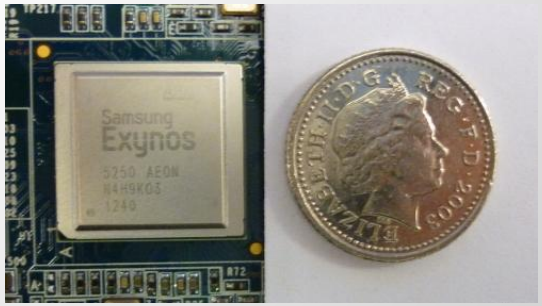
Portable Mapping of Data-Parallel Programs to OpenCL for Heterogeneous Systems

Dominik Grewe, Zheng Wang,
Michael O'Boyle

CGO 2013, Shenzhen, China
26 February 2013

Motivation

- Heterogeneous Computing has become mainstream
  - OpenCL as industry-wide standard
- High Performance Computing
  - dedicated GPUs
- Desktop/Mobile Computing
  - integrated GPUs
  - System-on-Chips



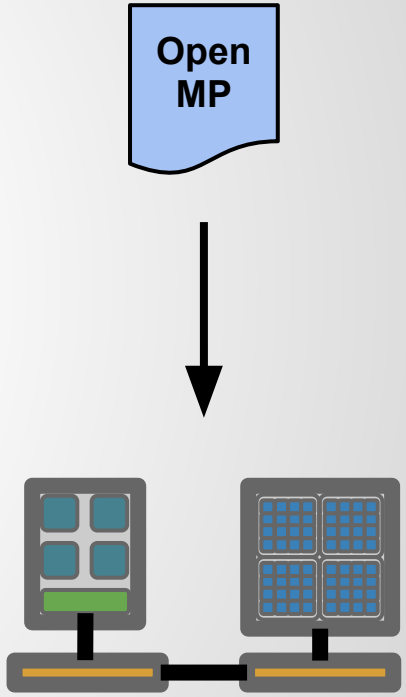
Two main challenges

- Task Mapping
  - Selecting the most suitable processor for a given task.
  - Partitioning workloads across processors.
  - Dealing with resource contention.

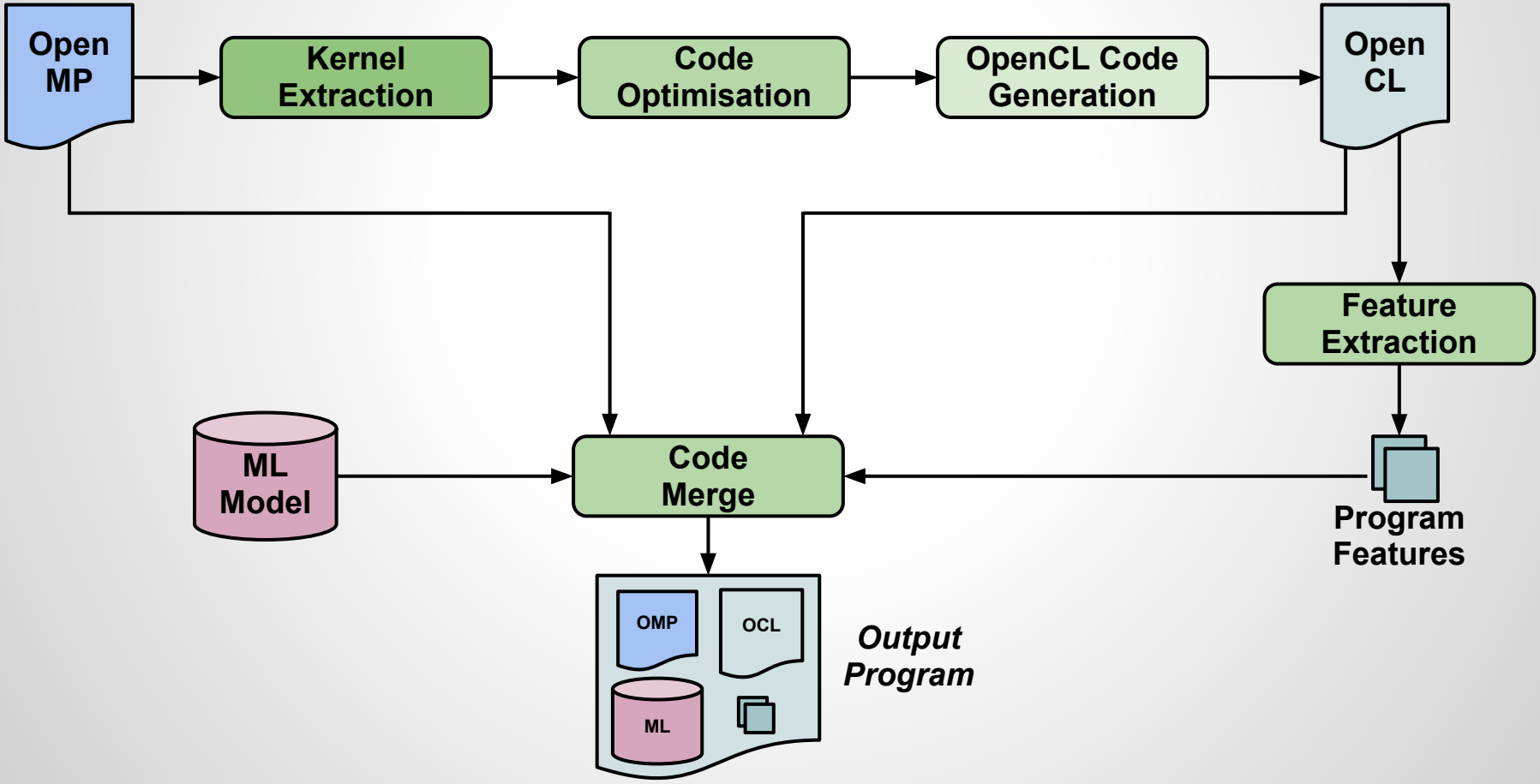
- Code Generation & Tuning
  - Generate low-level code from high-level languages.
    - OpenCL as intermediate representation
  - Optimize code for specific target architectures.
    - Data layout transformations
    - Parallelism mapping
    - ...

Mapping Data-Parallel Programs to OpenCL for Heterogeneous Systems

- OpenMP loop parallelism
- Generate efficient OpenCL code
  - optimize for GPU
- Pick target device
  - at runtime
  - using static & dynamic code features

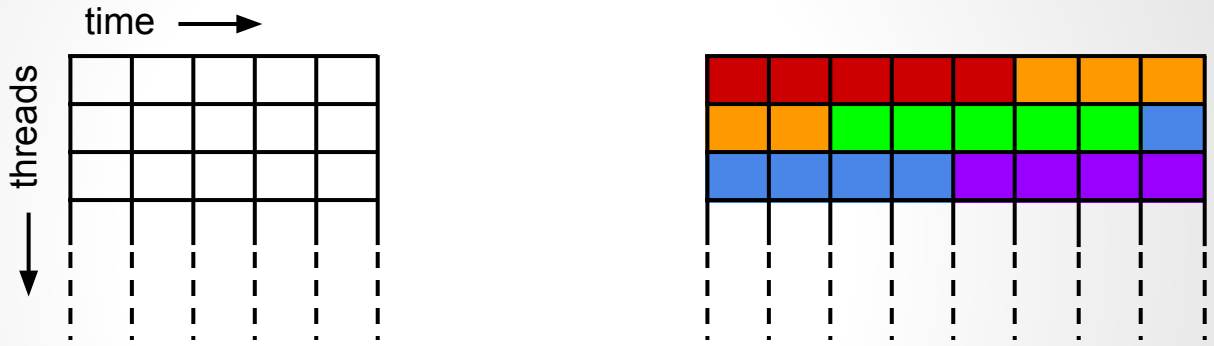


Mapping Data-Parallel Programs to OpenCL for Heterogeneous Systems

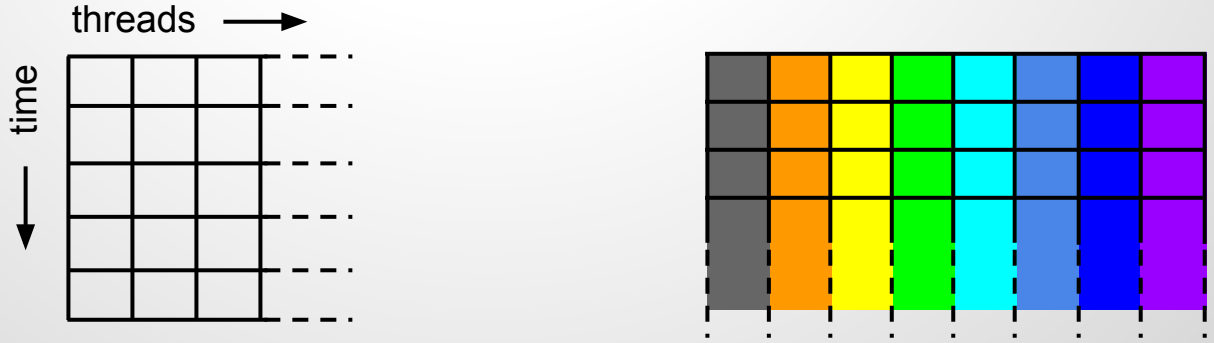


Optimising Memory Accesses

- CPU: *intra*-thread locality



- GPU: *inter*-thread locality
  - consecutive threads access consecutive data

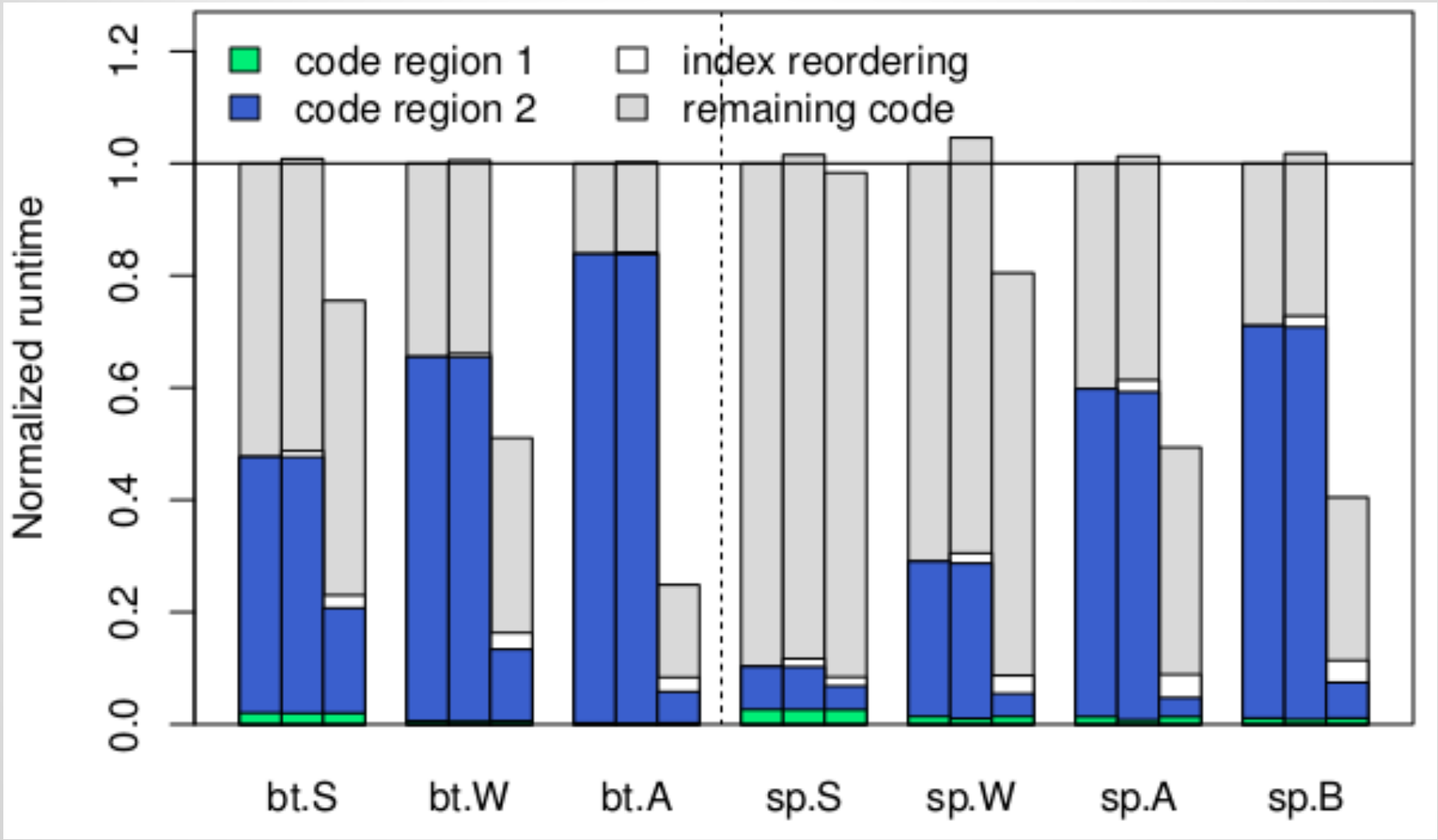


Dynamic Index Reordering

- Rearrange data dynamically at runtime.
  - A globally optimal data layout does not always exist.

- May not always be beneficial:
  - Cost: Data transformation
  - Benefit: Improved memory accesses

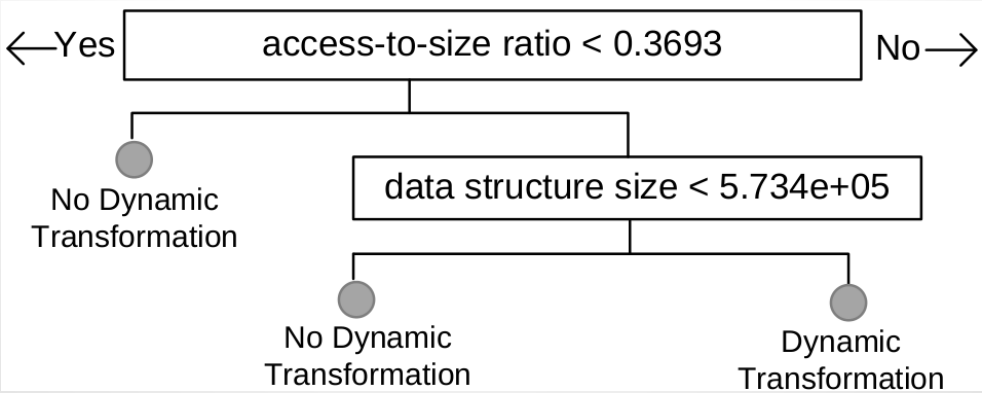
Dynamic Index Reordering



Dynamic Index Reordering

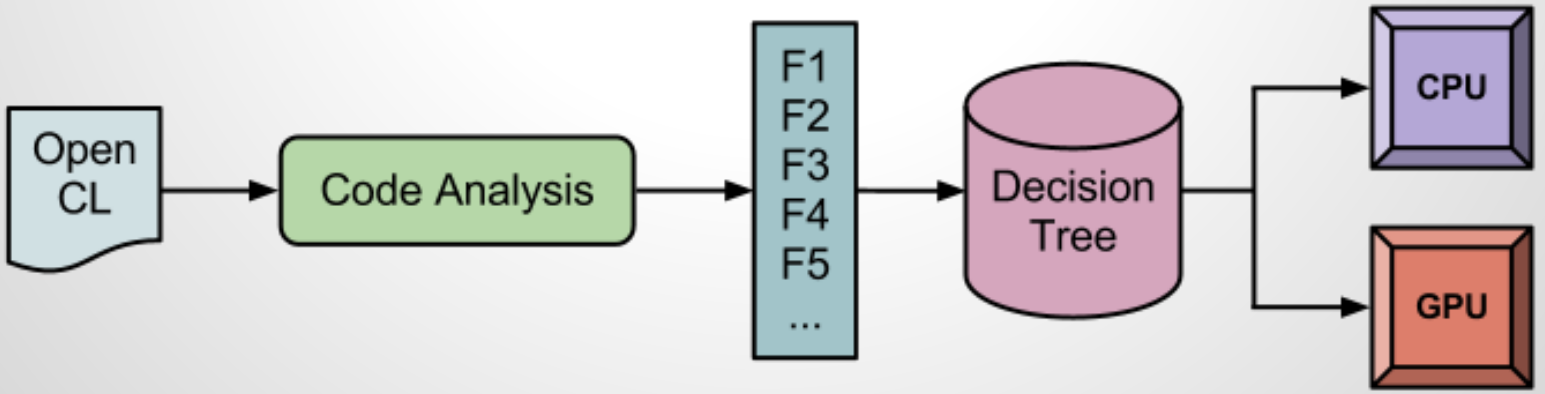
- Data-driven heuristic to decide when the transformation is beneficial.
  - size of data structure
  - #accesses to data structure

- using micro-benchmarks for training

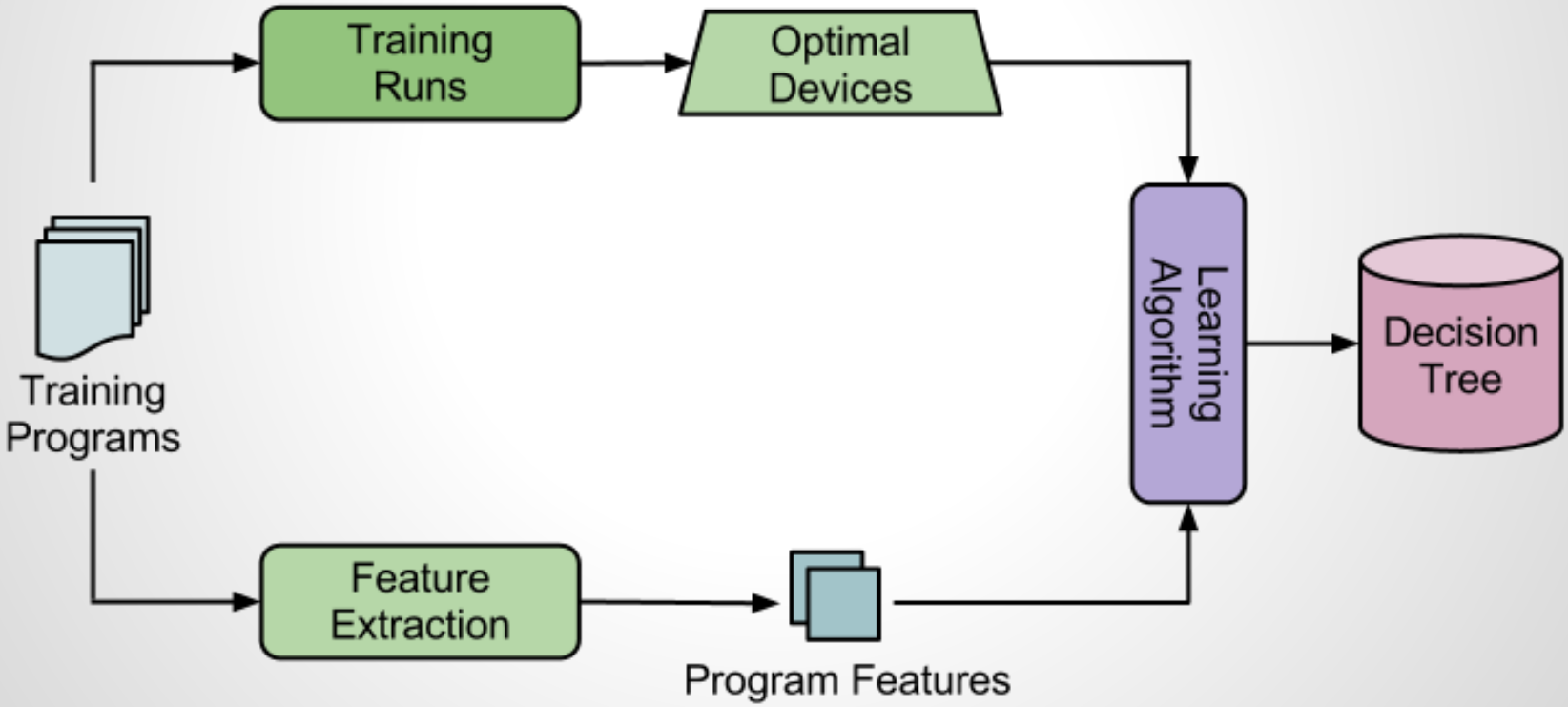


Predicting the Mapping

- Predict for the *whole program* whether to run parallel sections on CPU or GPU.
  - Binary decision tree classifier.
- Based on static code features.
  - Instantiated at run-time.
  - Aggregated across all parallel regions.



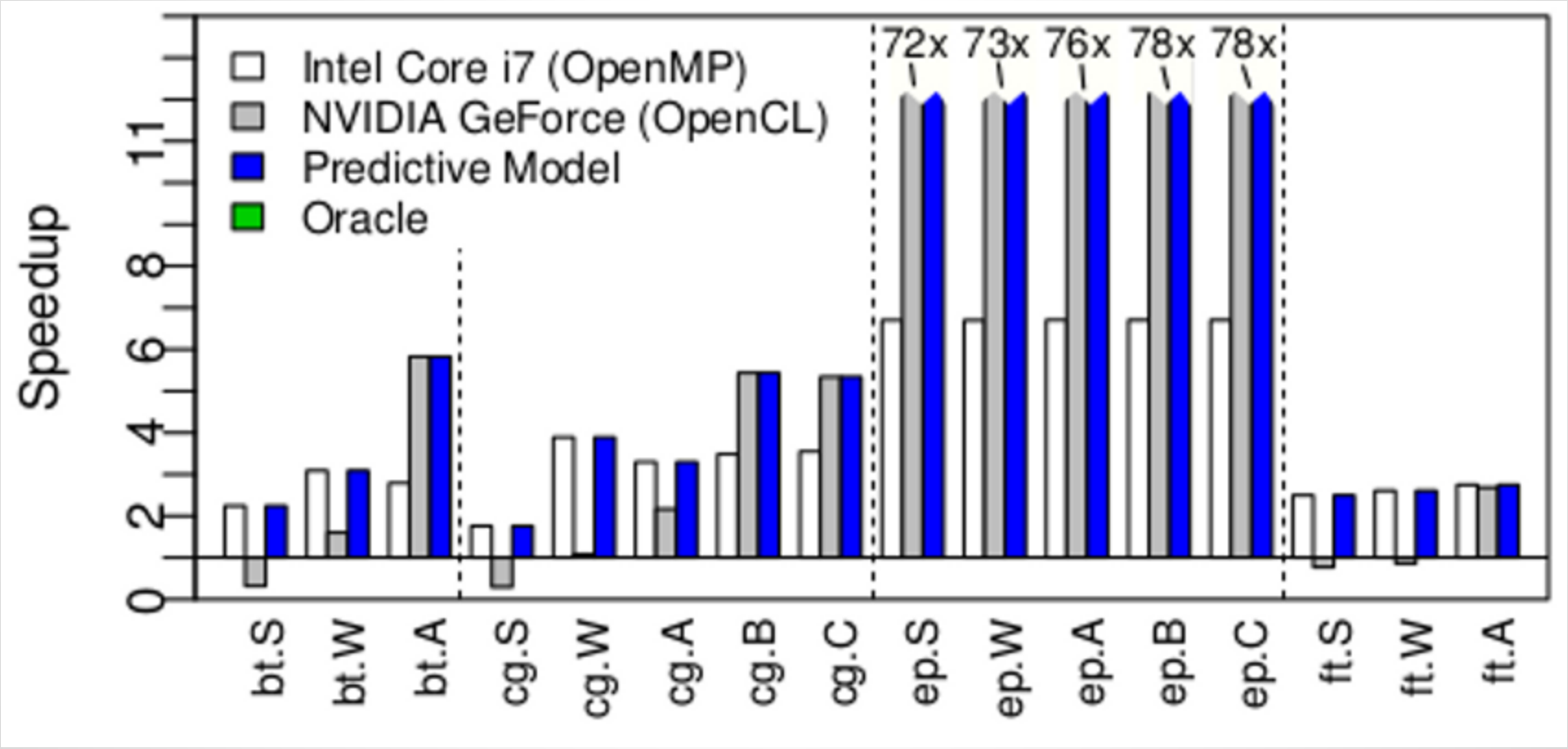
Creating the Decision Tree



Experimental Methodology

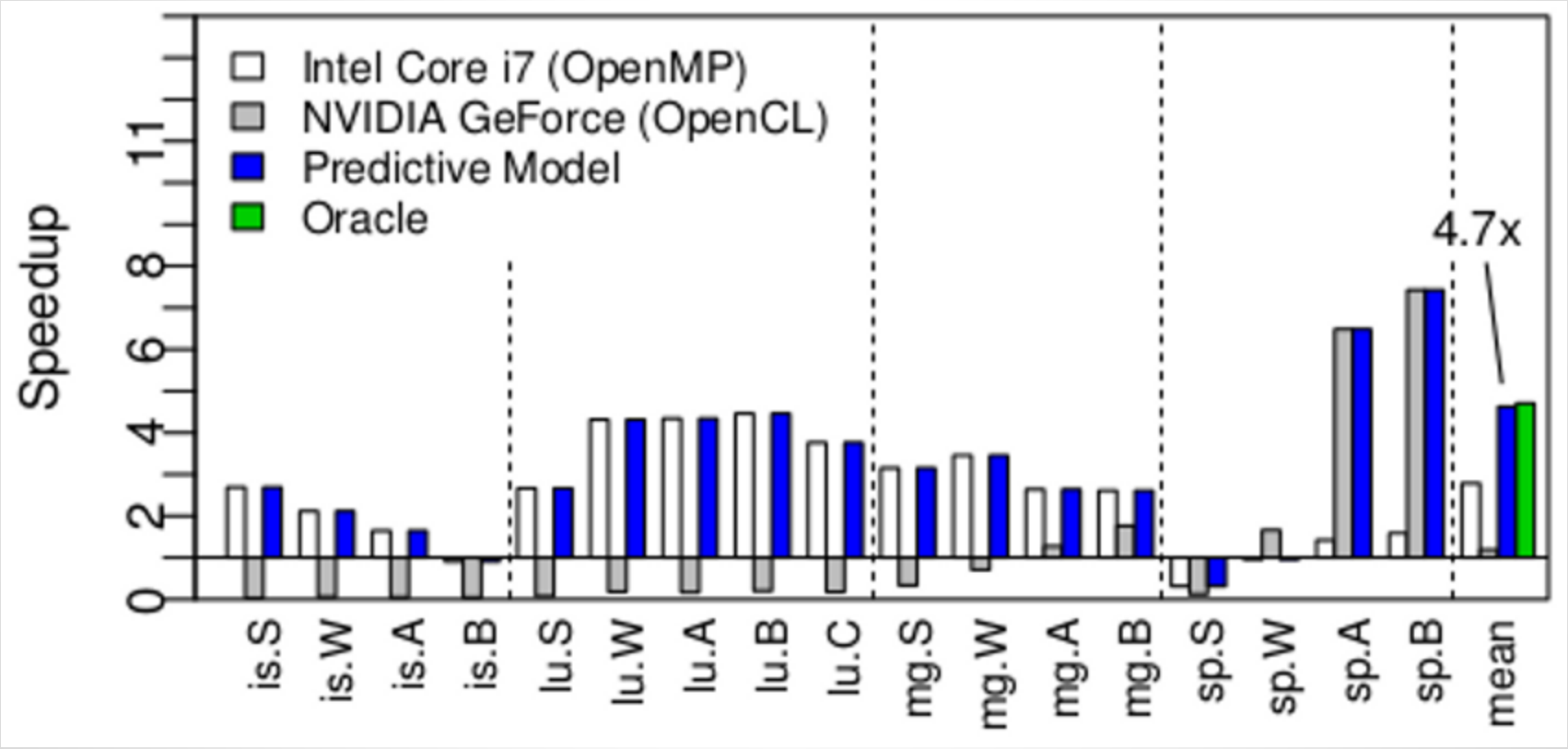
- Platforms
  - Intel CPU + NVIDIA GeForce
  - Intel CPU + AMD Radeon
  - AMD Llano APU
  - Intel IvyBridge
- NAS parallel benchmark
  - full suite of 8 benchmarks
- Comparison
  - closest related work: OpenMPC (Lee et al.)
  - hand-written OpenCL implementation (Seo et al.)

Performance Evaluation



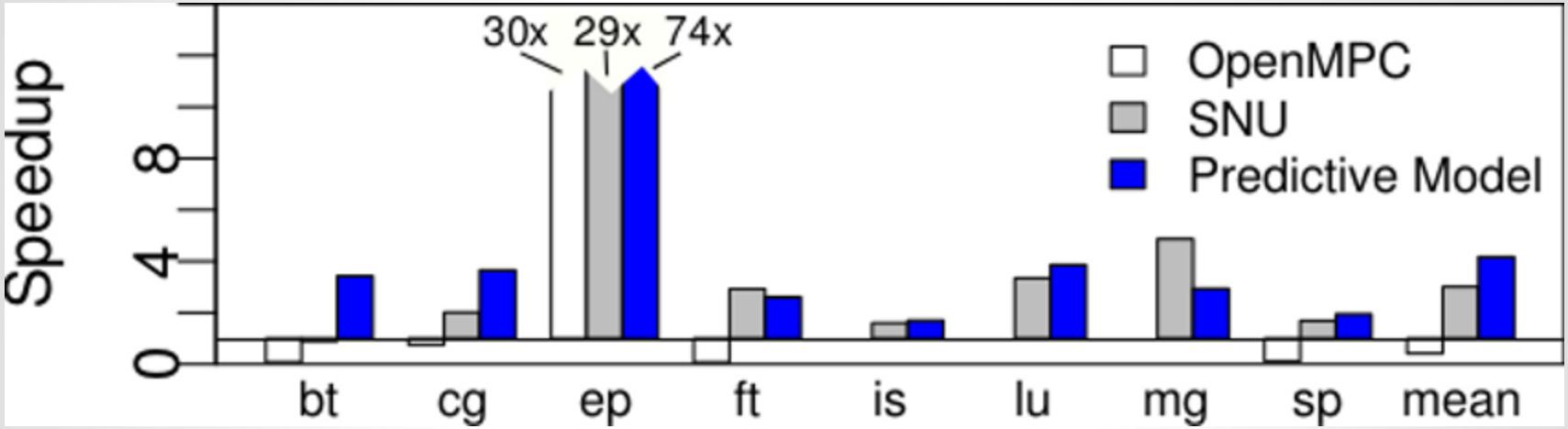
Intel Core i7 + NVIDIA GeForce GTX 580

Performance Evaluation (2)



Intel Core i7 + NVIDIA GeForce GTX 580

Comparison to State-of-the-Art



- SNU: hand-coded implementation
  - Seo et al. [IISWC 2011]
- OpenMPC: OpenMP to CUDA
  - Lee et al. [PPoPP 2009]

Mapping Parallel Programs to Heterogeneous Systems

- Mapping Tasks to Devices
  - Machine-learning model (decision tree) using code features.

- Generating and optimizing device code
  - Generate OpenCL from OpenMP parallel loops.
  - Data transformations for good GPU performance.

- Results
  - 1.67x speedup over original OpenMP code.
  - 1.63x speedup over hand-coded OpenCL code.