ParTeCL – Parallel Testing Using OpenCL

Vanya Yaneva Ajitha Rajan Christophe Dubach

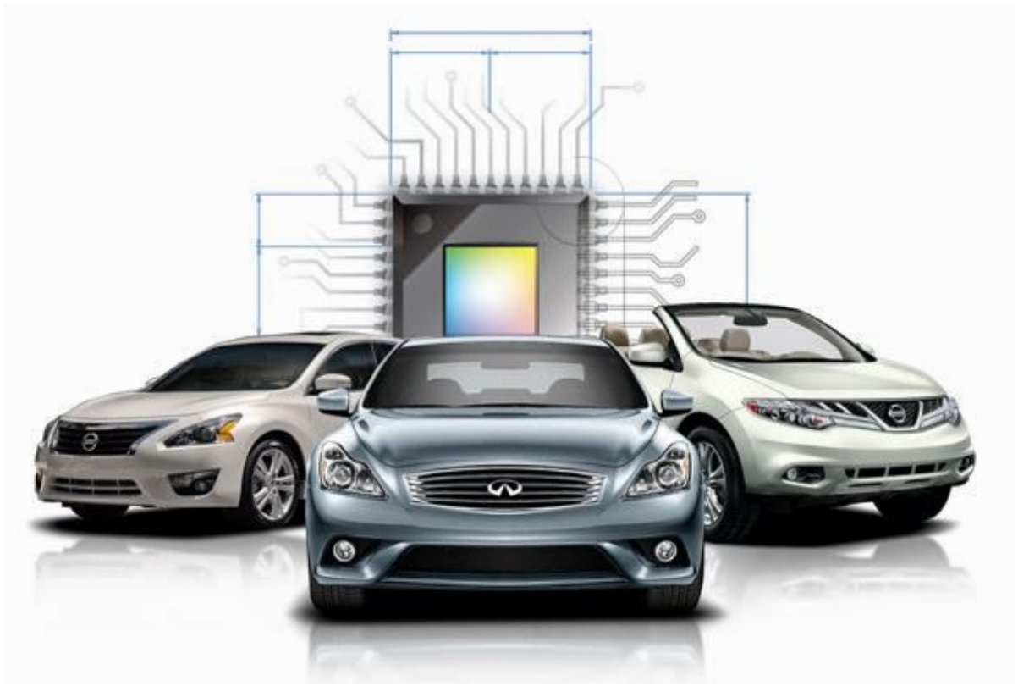
Motivation

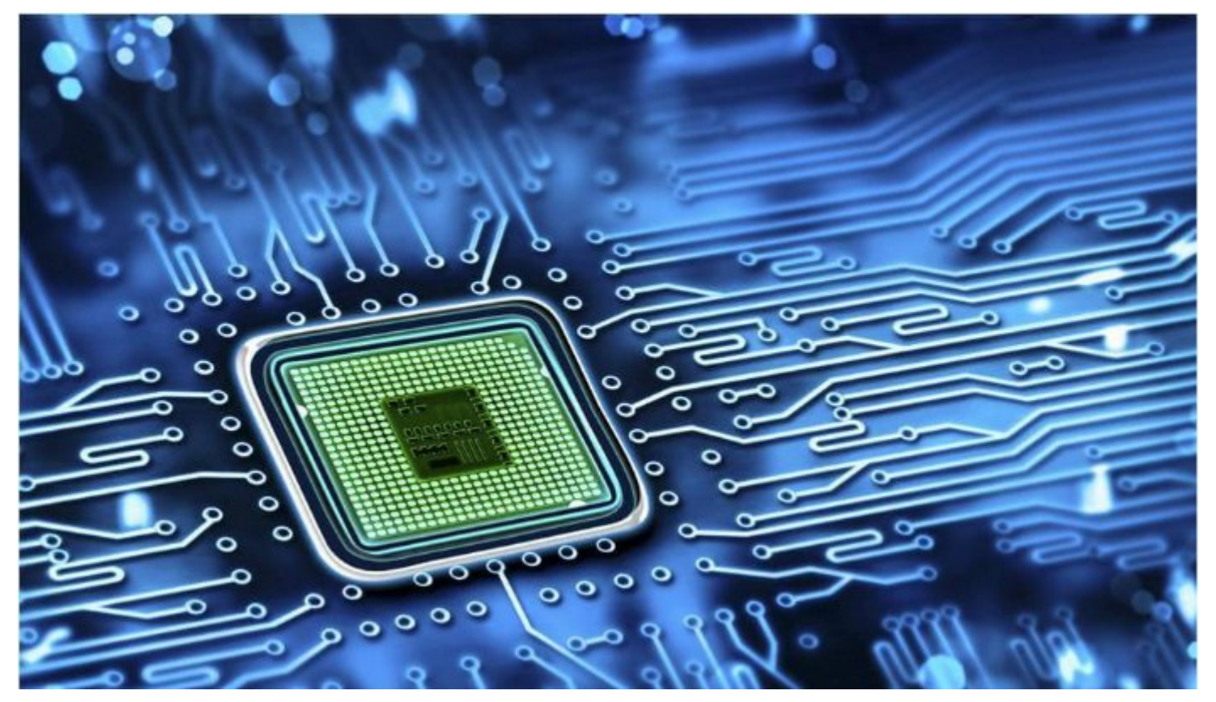
Background:

Software is ubiquitous and its correctness is critical.

Problem:

Functional software testing is crucial, but **extremely time consuming.**





Proposed solution:

Execute test cases in parallel on the GPU threads.

Benefits

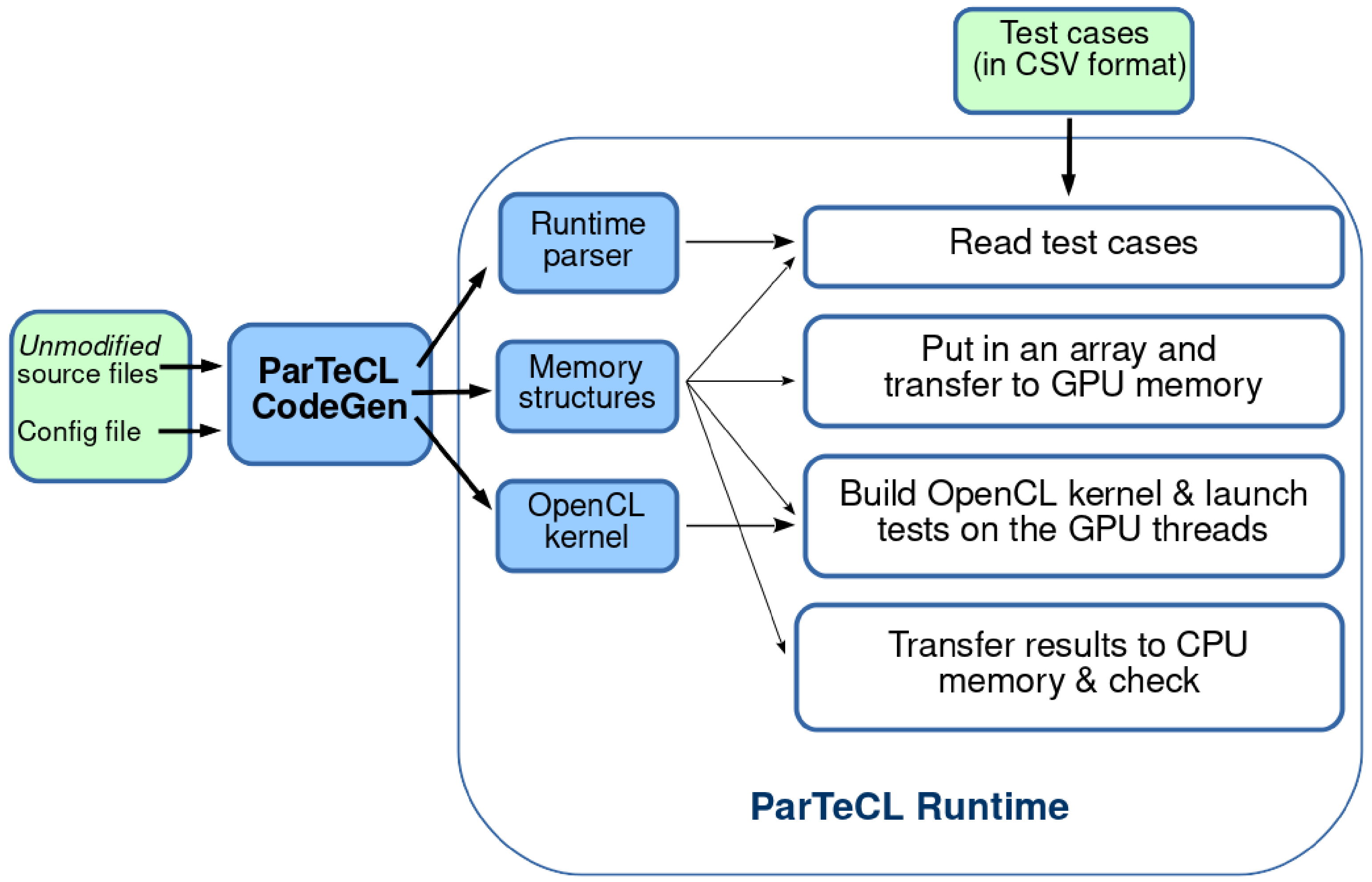
- **Reduced:**
  - testing time
  - cost of testing infrastructure & maintenance
  - energy consumption
- **Increased**:
  - speed of testing
  - frequency of testing
  - fault finding

Approach

Implemented two systems to automate test execution on the GPU:

- **ParTeCL CodeGen:** generates OpenCL from the source code of the tested program
- **ParTeCL Runtime:** launches the test cases on the GPU threads

Target: Embedded C programs



Test cases (in CSV format)

Unmodified source files

Config file

ParTeCL CodeGen

Runtime parser

Memory structures

OpenCL kernel

Read test cases

Put in an array and transfer to GPU memory

Build OpenCL kernel & launch tests on the GPU threads

Transfer results to CPU memory & check

ParTeCL Runtime

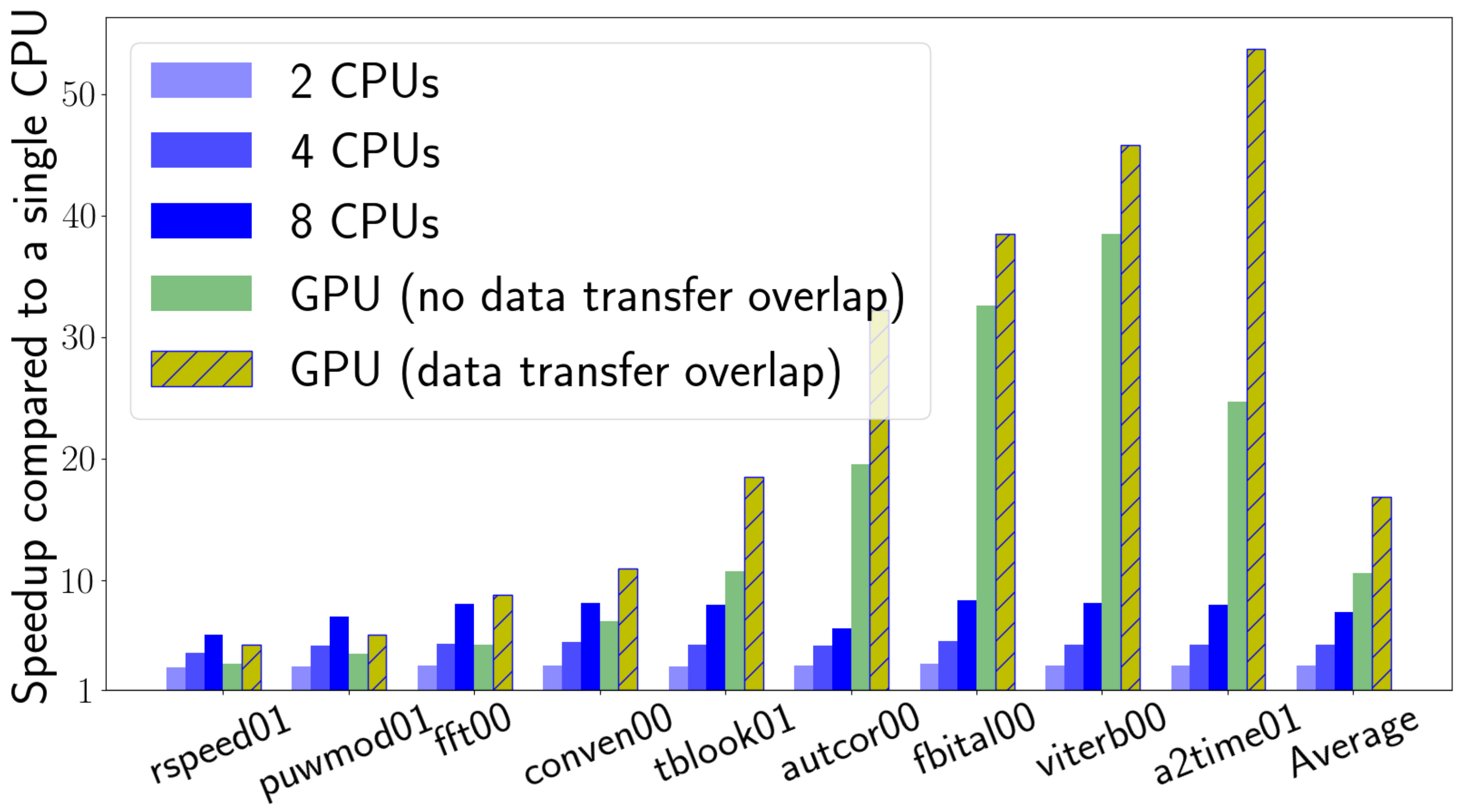
Results

Subjects: 9 programs from the **EEMBC** benchmark suite

Hardware: GPU: NVidia Tesla K40m
CPU: Intel(R) Xeon(R), 8 cores

Results:

- **Speedup up to 53x (average 16x)** compared to a single CPU
- **Correctness verified:** testing results on the GPU are the same as on the CPU



Limitations & Future work

1. **Control-flow divergence limits performance:** Group test cases based on control-flow paths
2. **Not all C features are readily supported for compilation on the GPU:** Implement code transformations in ParTeCL Codegen to handle those.
3. **Perform empirical evaluation on additional benchmarks.**

Links

- **github.com/wyaneva/partecl-codegen**
- **github.com/wyaneva/partecl-runtime**
- **github.com/wyaneva/clClibc**

THE UNIVERSITY *of* EDINBURGH
informatics

EPSRC Centre for Doctoral Training in
Pervasive Parallelism