Phoenix: Language-Driven Auto-Vectorisation

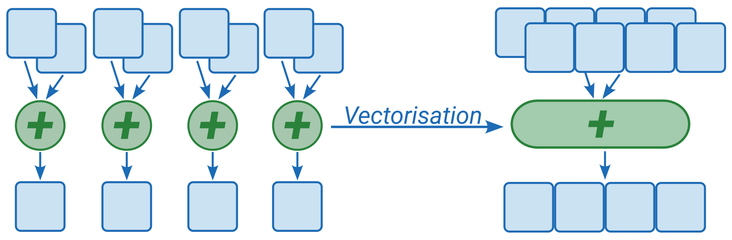
Jonas Sys
jonas.sys@ugent.be
Ghent University and Vrije Universiteit Brussel

Elisa Gonzalez Boix
egonzale@vub.be
Vrije Universiteit Brussel

Christophe Scholliers
christophe.scholliers@ugent.be
Ghent University

1. Research Context

Modern CPUs are equipped with SIMD units, allowing a single CPU core to evaluate the same instruction on multiple data values in parallel. Even commodity hardware is able to achieve significant performance improvements by leveraging data-parallelism.



Branching control flow obscures data-parallelism in scalar code

The problem: manual vectorisation is very complex, and existing tools fail to provide guarantees for efficiently exploiting SIMD units.

2. Existing Approaches

Manual vectorisation: guaranteed performance at extreme complexity

Automatic (auto-)vectorisation: no developer burden, but also no performance guarantees

Language extensions: low developer burden, but minor guarantees to performance and correctness (data race conditions)

3. Phoenix

Phoenix is a data-parallel programming language, designed to allow safe, efficient vectorisation while still closely resembling scalar code by employing restricted vector loops.
All restricted vector loops are subject to these three restrictions:

Iteration independence: to eliminate loop-carried dependencies, vector loops are not allowed to modify non-loop-local variables.

Loop completion: to be able to pre-compute the loop-trip-count, vector loops cannot early exit (i.e., no break or return).

Index-free loop operation: reading masked-out elements introduces undefined behavior. Phoenix prohibits arbitrary indexing in vector loops.

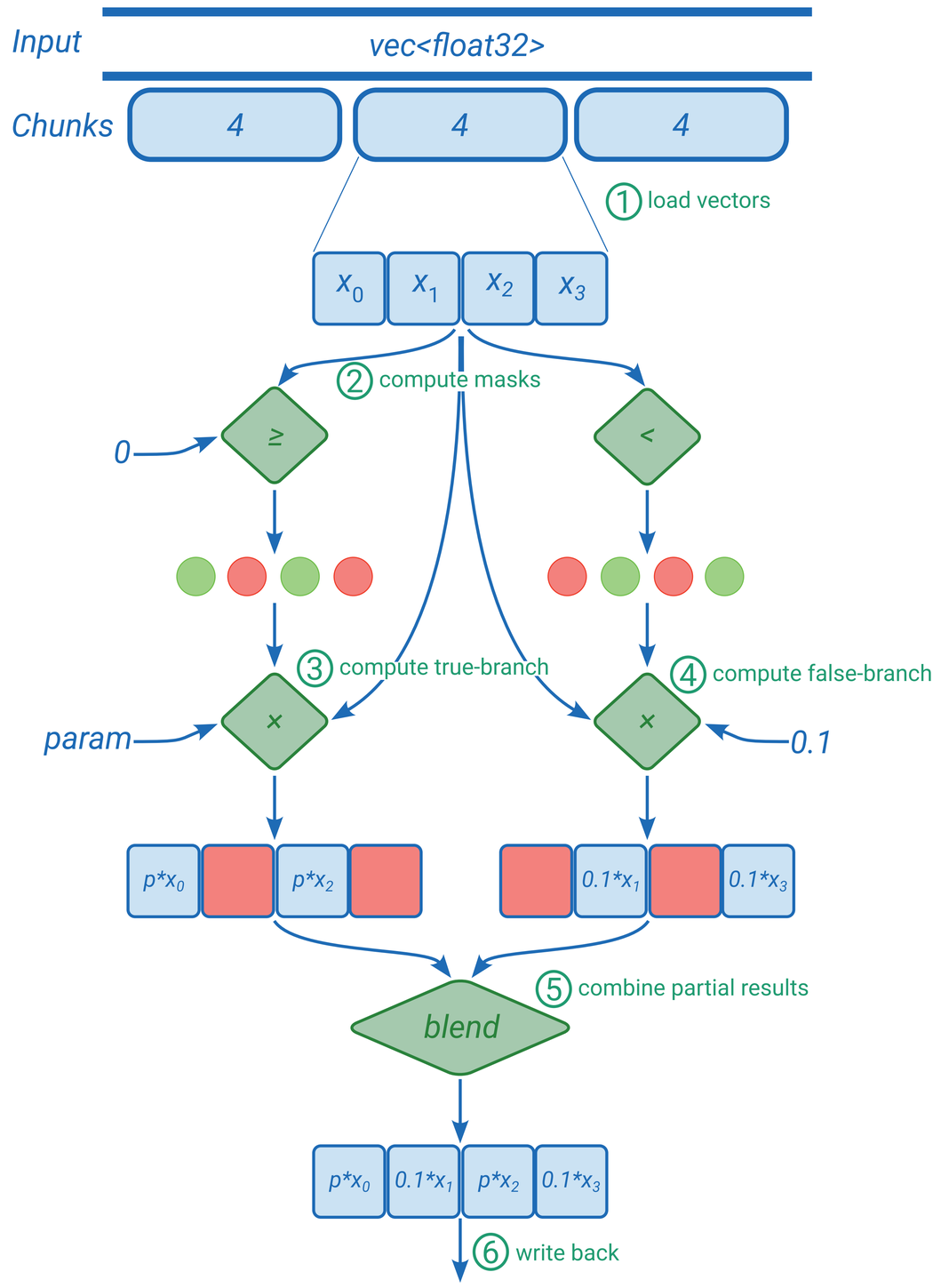
These three restrictions guarantee safe and correct vectorisability

Example: leaky scaled ReLU.

```
void leaky_relu(vec<float32> &xs, float32 param) {
  foreach x in xs {
    if (x ⩾ 0.0f) continue with param * x;
    else continue with 0.1f * x;
  }
}
```

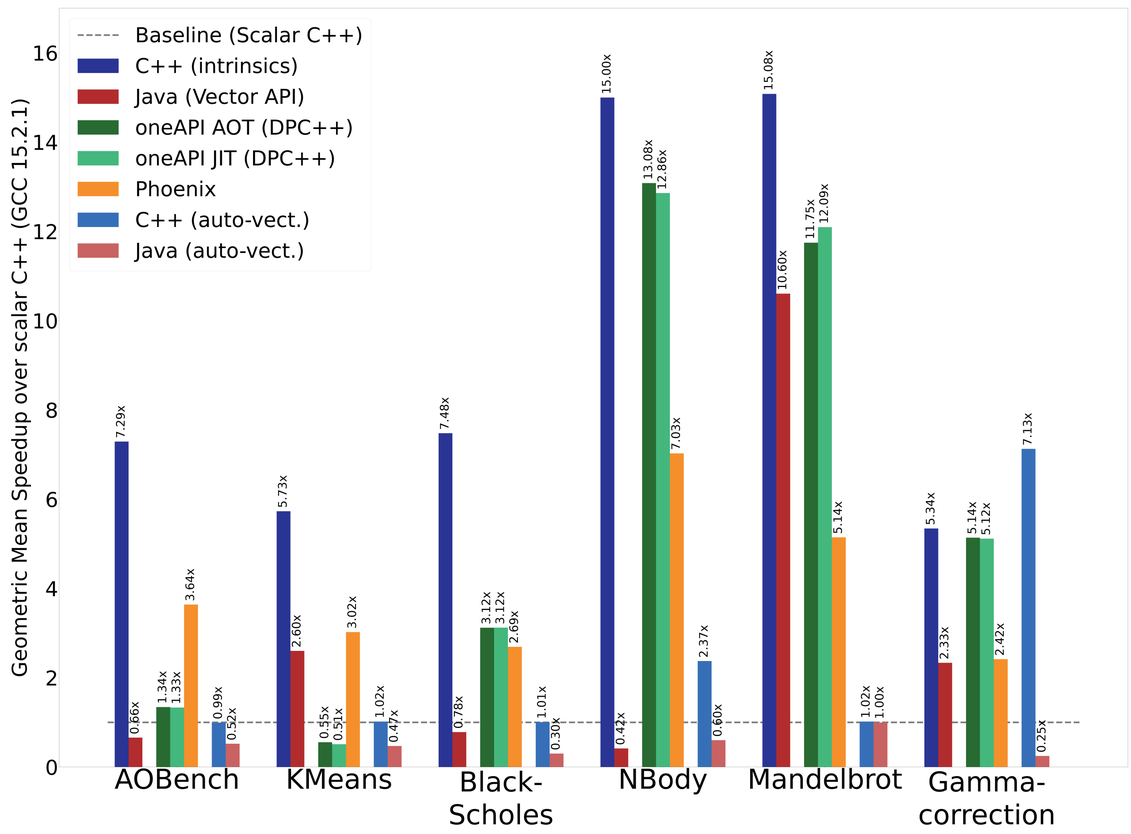
Phoenix's design eliminates compiler guesswork, guaranteeing vectorisability

4. Simplified Vectorisation Steps



5. Evaluation

We evaluated Phoenix through a benchmark suite with kernels from various application domains, showing that its expressivity is not hampered by its restrictions.



Complex control flow does not inhibit Phoenix's vectoriser.

GHENT UNIVERSITY FACULTY OF SCIENCES

VUB VRIJE UNIVERSITEIT BRUSSEL