Compilation of a DSL based on vectorial circuit to SIMD optimized code

Efficient Data Processing

Streaming data processing is a crucial approach that focuses on traversing data to extract pertinent information. Applications ranges from network packet manipulation to analysing DNA. Modern data-processing tools heavily depend on efficient implementations that harness hardware acceleration to achieve high performance. This acceleration can sometimes be achieved through automatic compilation, but frequently demands expert developers to craft optimizations by hand.

One critical facet of this optimization process involves SIMD optimization, where data is packed into chunks and processed with minimal branching in the code, often using bit vector operations. These optimizations are at the core of numerous well-known software applications, such as regular expression matching in tools like ripgrep, JSON parsing in libraries like SimdJSON, and even fundamental operations like string encoding and decoding (Unicode parsing). Developing these optimizations requires a broad skill set and is a testament to the expertise of programmers worldwide.

Designing VIR: an intermediate representation of vectorial programs.

During this PhD, we will explore the design and implementation of VIR, an intermediate representation of vectorial programs heavily influenced by synchronous programming, high-performance compilation of array languages and vectorial circuits.

The end goal is to have a machine and optimization friendly formal representation of computation relying heavily on SIMD accelerations.

Some cases studies have already been performed in the context of various experiences, coming from early-stages internships in which the premises of complete toolchain to evaluate simd solutions has been designed; or a complete project, vizitig, which proposes simd implementations of programs analyzing DNA strings. Th core of our specialized language will benefit from these experiences.

In the context of this PhD, the student will:

- make an extensive bibliography of existing approaches for compiling programs into vectorial code, focusing on intermediate representations to represent parallelism at different abstraction levels;
- study the specificities of different vectorial targets, especially from the circuit complexity point of view;
- make different propositions as VIR as intermediaire representation;
- contribute to the compilation stack inside the project, focusing on back-ends.

Location

This PhD will be co-advised by Laure Gonnord (Unversity Genoble Alpes) and Charles Paperman (Université de Lille) and will be located at Lille.

Candidate profile

The candidate should ideally be familiar with formal approaches in programming language design, notably type systems, semantics, and logic. From the practical point of view, a basic experience in software programming and usage of collaborative tools such than git. This PhD strongly relies on the fact that practical implementation should have strong theoretical foundations and that further refinements of the theory should get inspiration from the practical side. We expect the candidate to agree with this philosophy.

Bibliography

1. Parsing Gigabytes of JSON per Second
2. Supporting Descendants in SIMD-Accelerated JSONPath
3. Validating UTF-8 in less than one instruction per byte
4. Hyperscan: A Fast Multi-pattern Regex Matcher for Modern CPUs
5. Vectorial languages and linear temporal logic.
6. An algebraic approach to vectorial programs