Presentation Abstract

PelemayFp: An Efficient Parallelization Library for Elixir Based on Skeletons for Data Parallelism

Susumu Yamazaki[1,a)]

Presented: March 17, 2021

In this presentation, we propose Pelemay Fast Parallel map (PelemayFp), which is a library to parallelize Elixir code, efficiently, based on skeletons for data parallelism. PelemayFp is implemented using only Elixir, like Flow, which is a library of previous works. In Flow, the order of the list after computation is not guaranteed, while in PelemayFp, the order of the list is guaranteed because it is sorting while collecting and merging. On the other hand, Pelemay Super Parallelism (Pelemay), which we proposed, generates native code using SIMD instructions and calls it by NIFs, which is one of FFIs that Erlang provides, without performing multi-core parallelism, guaranteeing the order of the list. We evaluated the integer arithmetic performance by logistic mapping of PelemayFp alone, Pelemay alone, the combination of PelemayFp and Pelemay, Flow, and Enum, which is in the standard library of Elixir. When run on an Intel Xeon W-2191B CPU with 18 cores and 36 threads, the PelemayFp alone is up to 2.1 times faster than Enum. It is also faster than Flow without sorting. On the other hand, the combination of PelemayFp and Pelemay is up to 1.27 times faster than Enum. We also estimated the percentage of parallel execution in the entire code based on Amdahl's law. That of PelemayFp is 48–66 percent, while that of the combination of PelemayFp and Pelemay is 21–46 percent. Further analysis revealed that this experimental results can be explained by assuming that when calling native code from Elixir with NIFs, the part that is not executed in parallel increases by about 40 percent. Therefore, when generating native code including SIMD instructions and adopting the approach of parallelizing with Elixir for speeding up, it will be appropriate to incorporate a code optimization mechanism using SIMD instructions into the JIT, which will be released in the next major version of Erlang, or to use another FFI method, Port, instead of using NIFs.

This is the abstract of an unrefereed presentation, and it should not preclude subsequent publication.

[1] University of Kitakyushu, Kitakyushu, Fukuoka 808–0135, Japan
[a)] zacky@kitakyu-u.ac.jp