# **Key takeaways from the paper**

## USING AVX2 INSTRUCTION SET TO INCREASE PERFORMANCE OF HIGH PERFORMANCE

## COMPUTING CODE - Pawel Gepner

**Problem and Motivation**: The paper talks about how Intel Advance Vector Extensions 2(AVX2) is used in high performance computing. It also brings to light where the AVX2 might not be the most effective way to increase performance.

**What is AVX?** It’s a set of instructions for doing Single Instruction Multiple Data (SIMD) operations on intel architecture CPU’s.

**AVX features:**

* The 128-bit SIMD registers have been expanded to 256 bits.
* Three-operand, non-destructive operations have been added. Previous two-operand instructions performed operations such as A = A + B, which overwrites a source operand; the new operands can perform operations like A = B + C, leaving the original source operands unchanged.
* A few instructions take four-register operands, allowing smaller and faster code by removing unnecessary instructions.
* Memory alignment requirements for operands are relaxed.
* A new extension coding scheme (VEX) has been designed to make future additions easier as well as making coding of instructions smaller and faster to execute.

**How AVX helps?**

SIMD instructions allow processing of multiple pieces of data in a single step, speeding up throughput for many tasks, from video encoding and decoding to image processing to data analysis to physics simulations. Intel AVX instructions work on Institute of Electrical and Electronics Engineers (IEEE)-754 floating-point values in 32-bit length (called *single precision*) and in 64-bit length (called *double precision*). IEEE-754 is the standard defining reproducible, robust floating-point operation and is the standard for most mainstream numerical computations.

Instructions often come in scalar and vector versions. Vector versions operate by treating data in the registers in parallel "SIMD" mode; the scalar version only operates on one entry in each register. This distinction allows less data movement for some algorithms, providing better overall throughput.

**Functionalities of AVX2:**

* integer data types expanded to 256-bit SIMD,
* bit manipulation instructions,
* vector to vector shifts,
* Floating Point Multiply Accumulate (FMA).

**Things to note:**

* Transition between AVX/AVX2 and legacy intel streaming SIMD extensions may result in latency penalties. This is because 2 instruction sets operate on different level of YMM registers the hardware during transition must first save and then retrieve the contents of YMM register which penalizes each operation with several tens of additional cycles. This can be solved by setting upper 128 bits to zero of YMM with VZEROUPPER instructions. Another way to solve this is by converting legacy SSEx instructions to AVX/AVX2-128-bit instruction.

**Where FMA fails:**

Calculating x = a \* b + c \* d + e \* f + g \* h on FMA would take higher number of cycles in parallel. Paper does not justify why this is so. So, it is good for smaller add and multiplication sequences.

**Conclusion:**

The AVX2 code version utilizing FMA is 81% faster than version operating on the AVX instructions set and 3 times more powerful than code, which is relying on the SSE 4.2 instruction set architecture only! Latency of AVX2 is reduced to lesser cycles compared to FMA as AVX2 has 2 sets of FMA implemented on it. The major improvement on AVX2 is due to double the number of buffers as it functions on 256-bit integer operations, as opposed to 128-bit integer on AVX.

AVX2 has some negative effect on frequency and turbo scaling. But the global performance is significantly greater than non AVX instructions even when the processor is operating at a slightly lower frequency. Intel’s Turbo boost provides opportunistic frequency increases based on workload, number of active cores, temperature, power and current. It is very important to notice that to achieve this level of performance increase, the use of MKL (Math Kernel Library) is essential as it simplifies the process of optimization.

**References:**

1. GEPnER, P. (2017). Using AVX2 instruction set to increase performance of high performance computing code. *Computing and Informatics*, *36*(5), 1001-1018.
2. <https://software.intel.com/content/www/us/en/develop/articles/introduction-to-intel-advanced-vector-extensions.html>
3. <https://scc.ustc.edu.cn/zlsc/tc4600/intel/2018.1.163/mkl/ps2018/get_started.htm>