SIMD Instructions outside and inside Oracle 12c (12.1.0.2)

Laurent Léturgez
PREMISEO
Lille - France

Keywords:

SIMD, CPU, register, vector, instruction, assembly, In Memory, Compression Unit, Compression, function, C, debugger, tracing, event, optimizer.

Introduction

SIMD extensions have been used since early 70's in vector programming. They are mainly CPU specific and not Oracle specific. Nevertheless, they are used by Oracle and especially by In-Memory option to perform operations on Oracle In-Memory Compression Units. Oracle exploits SIMD extensions to perform data parallelism without concurrency.
In this presentation, the data structure (vectors), CPU Registers, SIMD instructions and how they are used outside Oracle will be presented. As a consequence, the combination of both (data structure and instructions) can significantly reduce the number of operations. During this part, code samples written in C will be presented and executed to demonstrate it.
Then, a focus will be made on SIMD instructions inside Oracle and how it uses them: use of specific libraries and Oracle kernel components that should use SIMD instructions.
The GNU Linux debugger helps in this way to detect internal procedures usage that are based on SIMD instructions. It helps to establish the link between SIMD instructions and Oracle operations like filters and aggregates. Then, it will be shown how SIMD instructions are involved in performance improvement.

SIMD Outside Oracle: What is a SIMD Instruction?

SIMD stands for Single Instruction Multiple Data. With these instructions, it is possible to process multiple data with only one CPU Instruction.
This feature is possible because SIMD is based on specific registers located directly in the CPU; these registers are processed with a set of specific CPU instructions.
Everything is located inside the CPU, that's why SIMD are not Oracle specific but CPU specific.
Many different SIMD instructions exist and each CPU provider delivers his chip with its own SIMD instruction: IBM Power uses Altivec instructions, Sparc chips use VIS instructions etc.
This presentation will mainly talk about SIMD instructions used by Intel x86 chips.

CPU registers are wider than traditional CPU registers (RSI, RDI, R8 etc.): 128 bits and up to 512 bits wide and they can contain many data in one register, which is different than traditional ones.

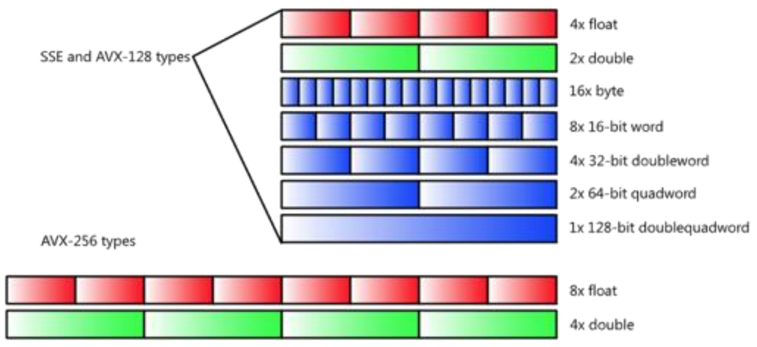


Illustration. 1: SIMD registers with different possible combinations to store many data into one register.

To illustrate how a SIMD instruction can reduce the number of CPU instructions, a basic example of a sum on an integer array executed in a scalar way and in a SIMD way is shown:

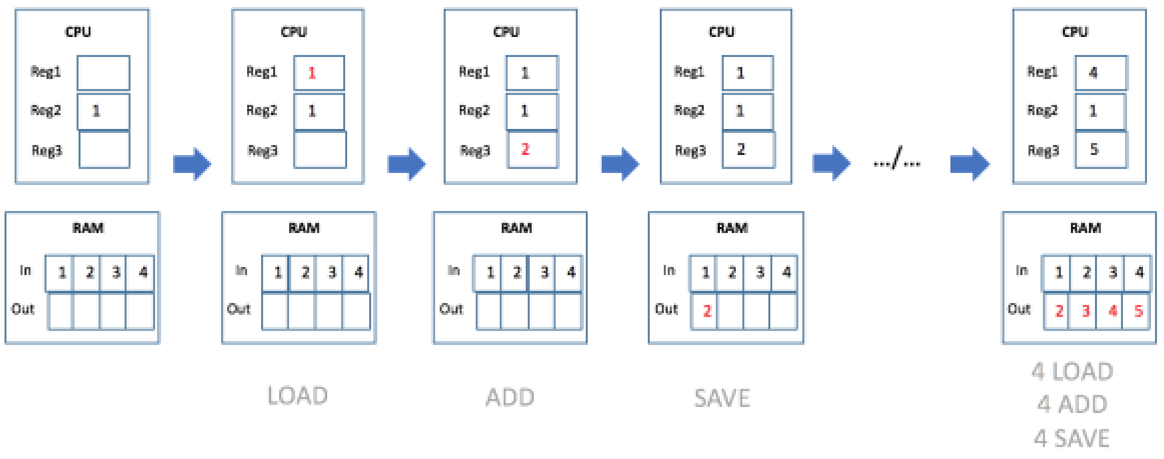


Illustration. 2: How a sum is processed on an array using scalar operation

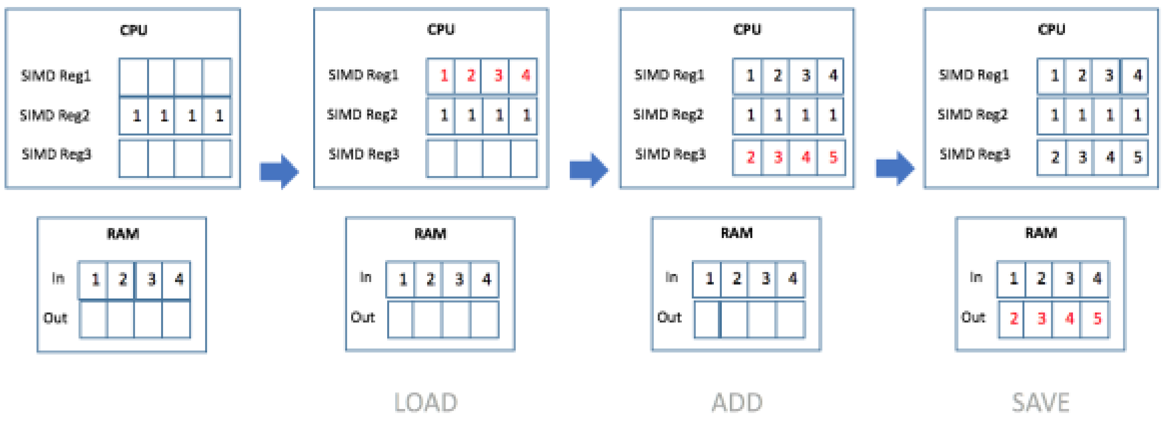


Illustration. 3: How is processed a sum on an array using SIMD instruction

These illustrations show that, on a four integer array, a simple sum executed by a SIMD instruction reduces the number of CPU operations (From 12 to 3 in this specific example).

As mentioned in the introduction, SIMD is not a very recent technology and it has evolved since the 70's. First Intel chip with "SIMD Inside" was the Pentium II with MMX extension because MMX is a SIMD instructions set.
MMX instructions were based on eight 64 bits registers and were extensively used by graphics and video applications. SIMD instructions are still extensively used by this kind of applications: for example, to reduce CPU operations when processing a picture with some image filters.

MMX processors have evolved since their release, and now SIMD instructions set have grown and CPU registers are 128 bits (SSE) or 256 bits wide (AVX), and keep on growing (up to 512 bits).

To develop software using SIMD instructions, Intel delivered an API (for C/C++) which is well documented in its "Intel Intrinsics Guide" available at this URL: https://software.intel.com/sites/landingpage/IntrinsicsGuide/

Some sample codes are available in the archive located at this URL: https://app.box.com/simdSampleC-2015

SIMD Outside Oracle: Will my application use SIMD?

Even if an application has been correctly developed with SIMD instruction, there are many other parameters that will determine if the application will use SIMD correct instructions set.
First of all is, the hardware, of course. If the CPU is not featured with the last SIMD instructions set … no chance to use it.

The other one is the hypervisor. Indeed, old hypervisor acts like a filter between virtual machines and hardware. And it will not present modern SIMD instructions set to the virtual machines.

Next one is the operating system. Some linux kernels (used in Redhat Linux 5 for example) haven't been built with the modern SIMD sets libraries or are unable to work with wide SIMD CPU registers.

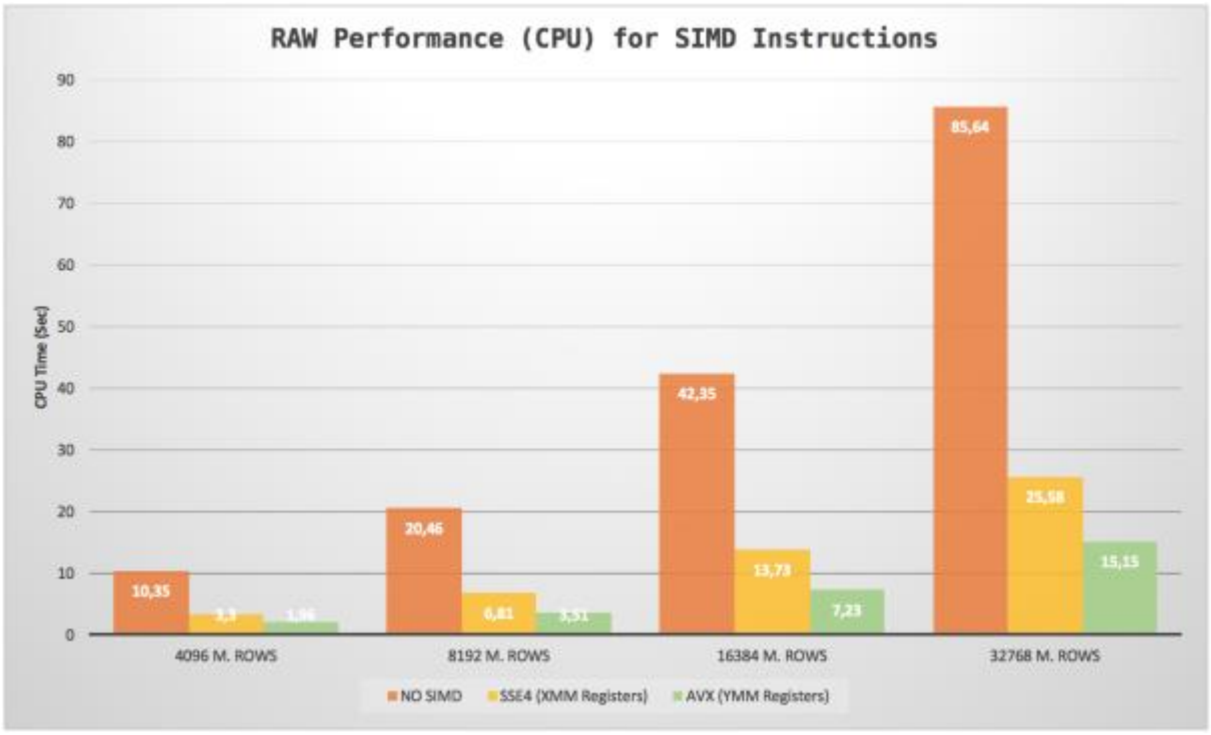
Another parameter which is very important is the compiler and the flags used to compile the code. Old compilers are not aware of the modern SIMD instructions set and even if they are, if compiler options or optimization flags are not correctly set, the assembly code can be rewritten (optimized) and will not use the correct SIMD instructions and/or registers.

SIMD Outside Oracle: Raw Performance

Raw performance has been measured with a very basic C program that performed sums on a huge array of data (up to 32768 Millions). The CPU time has been measured depending on the SIMD instructions.

The source code of this example can be freely downloaded. (https://app.box.com/s/ibmnbblpho4xtbeq2x8ir60nrk37208v). Please note that this code is very basic and only for demonstration purpose, real life offers more advanced code.

Results are presented below:



On the results, modern (and wider) SIMD registers usage reduce significantly the CPU time.

SIMD Inside Oracle: How SIMD instructions are used inside Oracle 12c

Inside Oracle, SIMD instructions are used with the "In Memory Column Store" Option (IM). The data structure (In Memory Compression Unit: IMCU) used by IM is very well suited for being used by SIMD Instructions.
Used in combination with IMCU pruning and their Storage Indexes, it gives Oracle very efficient filtering capabilities. But SIMD instructions offer the opportunity to process columns (mathematical operations etc.) at a very high rate.

Inside Oracle, there are some libraries used by the kernel to process data with SIMD instructions.
Analysis of these libraries showed that Oracle doesn't use last SIMD extensions in its kernel.
To complete this, and as there are a lot of parameters that enable a specific SIMD instructions set, analysing the Oracle kernel behaviour helps to determine how it uses SIMD from the hardware to the CPU register's usage.

For example, some tools can be used to check:

- Which SIMD extension is used by the CPU in the OS (/proc/cpuinfo)
- Which SIMD related library is used by Oracle (pmap)
- Which compiler and options are used to compile Oracle Kernel (readelf)
- How the SIMD registers are filled during SQL execution against IMCUs (gdb)

All of these tools and analysis will help to know what exact instructions set is used on the systems and what are their capabilities.

SIMD Inside Oracle: Tracing SIMD in Oracle 12c

Finally, to have a deeper analysis, Oracle tracing system (UTS: Unified Tracing System) can help to know more about how SIMD is used by Oracle 12c.

First, it can be used to trace Cost Based Optimizer (CBO) computation when it evaluates the cost of decompressing IMCU (It uses IM_Optimizer handle in UTS).

Once the statement is hard-parsed, data will be generated in the trace file. In this file, information available is:

- IMCU pruning ratio
- Compression Unit decompression costing.
  This concerns all the columns involved globally in the query execution.
- Predicate evaluation costing.
  This concerns every column involved in filtering predicates.

All this information can be found as a summary in global CBO trace file (10053 event or SQL_Costing_Event in UTS format).

Next point in tracing SIMD, is to use ADVCMP_DECOMP_HPK handle in UTS to trace specific SIMD functions usage in Oracle.

Positioning this event in the Oracle session will generate a trace file where these elements can be found:

- A detailed view of every IMCU processed. As IMCU are usually compressed, information about function and algorithm used by Oracle to process each IMCU is available.

  For every IMCU, processing details will be available and more specifically:
- Processing rates (in Millions rows per second). These rates are available for decompression operations and SIMD function executions.
- Decompression functions used and by consequence when possible, decompression algorithms.

SIMD instructions are executed inside decompression functions. When tracing SIMD functions, usage of decompression functions is mentioned, and consequently algorithms name appeared.
Analysis of different types of compression used to compress IMCU showed that many algorithms are used, and the chosen algorithm depends on many parameters like datatype, column compression unit size, column contents etc.

Oracle In Memory use a combination of many features that make it a powerful tool for analytics and column processing:

- A data structure: the In Memory Compression Unit
- Storage indexes and SIMD for a powerful filtering/processing engine
- Many compression algorithms.

Contact address:

Laurent Leturgez
PREMISEO
7, rue Jules Guesde
59320 EMMERIN - France

Phone:	+33 (0)6 60 99 70 46
Fax:	
Email	laurent.leturgez@premiseo.com
Internet:	http://www.premiseo.com