What is "vectorization"?

Asked 12 years, 3 months ago Active 4 months ago Viewed 127k times

261

Several times now, I've encountered this term in matlab, fortran ... some other ... but I've never found an explanation what does it mean, and what it does? So I'm asking here, what is vectorization, and what does it mean for example, that "a loop is vectorized" ?

vectorization simd auto-vectorization

114

Share Follow

edited Aug 11 at 19:46
Peter Cordes
269k 40 480 685

asked Sep 14 '09 at 15:07
Thomas Geritzma
5,707 6 23 19

2 @geoffspear The link seems to have been moved to en.wikipedia.org/wiki/Array_programming – I Like to Code Jul 19 '17 at 2:55

7 Answers

Active Oldest Votes

309

Many CPUs have "vector" or "SIMD" instruction sets which apply the same operation simultaneously to two, four, or more pieces of data. Modern x86 chips have the SSE instructions, many PPC chips have the "Altivec" instructions, and even some ARM chips have a vector instruction set, called NEON.

"Vectorization" (simplified) is the process of rewriting a loop so that instead of processing a single element of an array N times, it processes (say) 4 elements of the array simultaneously N/4 times.

(I chose 4 because it's what modern hardware is most likely to directly support; the term "vectorization" is also used to describe a higher level software transformation where you might just abstract away the loop altogether and just describe operating on arrays instead of the elements that comprise them)

The difference between vectorization and loop unrolling: Consider the following very simple loop that adds the elements of two arrays and stores the results to a third array.

```
for (int i=0; i<16; ++i)
    C[i] = A[i] + B[i];
```

Unrolling this loop would transform it into something like this:

```
for (int i=0; i<16; i+=4) {
    C[i]   = A[i]   + B[i];
    C[i+1] = A[i+1] + B[i+1];
    C[i+2] = A[i+2] + B[i+2];
    C[i+3] = A[i+3] + B[i+3];
}
```

Vectorizing it, on the other hand, produces something like this:

```
for (int i=0; i<16; i+=4)
    addFourThingsAtOnceAndStoreResult(&C[i], &A[i], &B[i]);
```

Where "addFourThingsAtOnceAndStoreResult" is a placeholder for whatever intrinsic(s) your compiler uses to specify vector instructions. Note that some compilers are able to *auto vectorize* very simple loops like this, which can often be enabled via a compile option. More complex algorithms still require help from the programmer to generate good vector code.

Share Follow

edited Sep 14 '09 at 17:01

answered Sep 14 '09 at 15:12

Stephen Canon
99.6k 18 175 260

18 What's the difference between this and loop unwinding/unrolling? – Jeremy Powell Sep 14 '09 at 16:28

3 Isn't it true that a compiler would have an easier job auto-vectorizing the unrolled loop ? – Nikos Athanasiou Jun 10 '15 at 23:35

2 @StephenCanon how can one check whether or not some lines have been vectorized? If one would use objdump, what would one look for in the output of objdump? – user1823664 Jun 9 '17 at 13:49

5 @Shuklaswag: vectorization is something that compilers can do for you, but it's also something that programmers explicitly do themselves. The OS is not involved. – Stephen Canon Sep 13 '17 at 20:39

1 @user1823664 SIMD instructions and registers should be present in the objdump. Example of

```
}
```

Vectorized approach:

```
for (i = 0; i < 1024; i+=4)
{
   C[i:i+3] = A[i:i+3]*B[i:i+3];
}
```

Share Follow

answered Sep 14 '09 at 15:13

Anders
11.3k 34 95 143

2 isn't that in essence same as Scalar approach? Your syntax and loop advancing is different , but underneath you are still multiplying it 4 times. But somehow it will be faster probably the CPU has instructions that does some trick called Vectorization. – mskw Aug 23 '17 at 3:30

3 Looks like I will answer my own question here. The syntax in the vectorization approach when the complier see that, it will translate it into optimized CPU instructions that multiplies vectors. Like SIMD. – mskw Sep 23 '17 at 15:20

17

Vectorization is used greatly in scientific computing where huge chunks of data needs to be processed efficiently.

In real programming application , i know it's used in NUMPY(not sure of other else).

Numpy (package for scientific computing in python) , uses **vectorization** for speedy manipulation of n-dimensional array ,which generally is slower if done with in-built python options for handling arrays.

although tons of explanation are out there , HERE'S WHAT **VECTORIZATION** IS DEFINED AS IN **NUMPY DOCUMENTATION PAGE**

Vectorization describes the absence of any explicit looping, indexing, etc., in the code - these things are taking place, of course, just “behind the scenes” in optimized, pre-compiled C code. Vectorized code has many advantages, among which are:

1. vectorized code is more concise and easier to read
2. fewer lines of code generally means fewer bugs
3. the code more closely resembles standard mathematical notation (making it easier, typically, to correctly code mathematical constructs)
4. vectorization results in more “Pythonic” code. Without vectorization, our code would be

littered with inefficient and difficult to read for loops.

Share Follow

edited May 3 '17 at 15:37

answered May 2 '17 at 4:34

bad programmer
551 4 9

13

It refers to a the ability to do single mathematical operation on a list -- or "vector" -- of numbers in a single step. You see it often with Fortran because that's associated with scientific computing, which is associated with supercomputing, where vectorized arithmetic first appeared. Nowadays almost all desktop CPUs offer some form of vectorized arithmetic, through technologies like Intel's SSE. GPUs also offer a form of vectorized arithmetic.

Share Follow

answered Sep 14 '09 at 15:09

Warren Young
38.1k 8 81 96

12

Vectorization, in simple words, means optimizing the algorithm so that it can utilize SIMD instructions in the processors.

AVX, AVX2 and AVX512 are the instruction sets (intel) that perform same operation on multiple data in one instruction. for eg. AVX512 means you can operate on 16 integer values(4 bytes) at a time. What that means is that if you have vector of 16 integers and you want to double that value in each integers and then add 10 to it. You can either load values on to general register [a,b,c] 16 times and perform same operation or you can perform same operation by loading all 16 values on to SIMD registers [xmm,ymm] and perform the operation once. This lets speed up the computation of vector data.

In vectorization we use this to our advantage, by remodelling our data so that we can perform SIMD operations on it and speed up the program.

Only problem with vectorization is handling conditions. Because conditions branch the flow of execution. This can be handled by masking. By modelling the condition into an arithmetic operation. eg. if we want to add 10 to value if it is greater then 100. we can either.

```
if(x[i] > 100) x[i] += 10; // this will branch execution flow.
```

or we can model the condition into arithmetic operation creating a condition vector c,

```
c[i] = x[i] > 100; // storing the condition on masking vector
x[i] = x[i] + (c[i] & 10) // using mask
```

this is very trivial example though... thus, c is our masking vector which we use to perform binary operation based on its value. This avoid branching of execution flow and enables

vectorization.

Vectorization is as important as Parallelization. Thus, we should make use of it as much possible. All modern days processors have SIMD instructions for heavy compute workloads. We can optimize our code to use these SIMD instructions using vectorization, this is similar to parrallelizing our code to run on multiple cores available on modern processors.

I would like to leave with the mention of OpenMP, which lets yo vectorize the code using pragmas. I consider it as a good starting point. Same can be said for OpenACC.

Share Follow

answered Dec 20 '18 at 17:05

Market Queue
521 5 7

7

By Intel people I think is easy to grasp.

> ***Vectorization is the process of converting an algorithm from operating on a single value at a time to operating on a set of values at one time***. Modern CPUs provide direct support for vector operations where a single instruction is applied to multiple data (SIMD).
>
> For example, a CPU with a 512 bit register could hold 16 32- bit single precision doubles and do a single calculation.
>
> 16 times faster than executing a single instruction at a time. Combine this with threading and multi-core CPUs leads to orders of magnitude performance gains.

Link https://software.intel.com/en-us/articles/vectorization-a-key-tool-to-improve-performance-on-modern-cpus

In Java there is a option to this be included in JDK 15 of 2020 or late at JDK 16 at 2021. See this official issue.

Share Follow

edited Aug 11 at 11:45

buhtz
7,377 11 55 105

answered Apr 11 '20 at 1:20

chiperortiz
4,471 7 42 71

-5

See the two answers above. I just wanted to add that the reason for wanting to do vectorization is that these operations can easily be performed in paraell by supercomputers and multi-processors, yielding a big performance gain. On single processor computers there will be no performance gain.

Share Follow

answered Sep 14 '09 at 15:13

Larry Watanabe
9,878 9 41 45

16 "On single processor computers there will be no performance gain": not true. Most modern processors have (limited) hardware support for vectorization (SSE, Altivec. etc. as named by stephentyrone), which can give significant speedup when used. – sleske Sep 14 '09 at 15:24

1 thanks, I forgot that parallelization can be done at that level as well. – Larry Watanabe Sep 15 '09 at 13:14