Ruby master - Feature #14328

SIMD vectorization

01/07/2018 04:13 PM - ahorek (Pavel Rosický)

Status:	Open
Priority:	Normal
Assignee:	
Target version:	

Description

Hello,
in order to make ruby faster, I'd like to propose an optional SIMD optimization for some cases. I want to target SSE2 which is available in all modern x86 processors. (Pentium 4, Athlon 64 and newer).

this is usually automatically handled by GCC during compilation time, but because of dynamic nature of ruby, redefinitions etc. It's very hard to preoptimize it before the actual execution.

use auto-vectorization provided by JIT (https://bugs.ruby-lang.org/issues/12589)

```
GCC can do that, but I'm not sure how reliable and effective it is today

Pros:
we don't have to do anything, let GCC do the job
bigger scope for optimizations

Cons:
slower compilation
```

- gcc docs:
- https://gcc.gnu.org/projects/tree-ssa/vectorization.html
- pypy has this feature implemented for some time now:
- https://morepypy.blogspot.cz/2015/10/pypy-400-released-jit-with-simd.html

specialize known bottlenecks by hand

```
Pros:
predictable performace
without increased compilation time

Cons:
code complexity
```

unfortunatelly using SIMD isn't for free, there's an overhead, it needs a large data set to be effective. It's useful mainly for math operations, sum, min, max, arrays, matrixes, string manipulations etc. There probably won't be any significant benefit for appliactions like Rails.

what do you think about it?

Related issues:

Related to Ruby master - Misc #16487: Potential for SIMD usage in ruby-core	**Open**

Associated revisions

Revision 7387c083 - 07/06/2018 01:56 PM - nobu (Nobuyoshi Nakada)

const_missing on private constants

- variable.c (rb_const_search): call #const_missing method on private constants, as well as uninitialized constants. [Feature #14328]

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@63871 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 63871 - 07/06/2018 01:56 PM - nobu (Nobuyoshi Nakada)

const_missing on private constants

- variable.c (rb_const_search): call #const_missing method on private constants, as well as uninitialized constants. [Feature #14328]

History

#1 - 01/11/2018 01:01 PM - naruse (Yui NARUSE)

I had tried to use SIMD in some parts.
But its performance improvement is limited.

Of course it can improve performance so much, but it is only in special use cases.
In usual Ruby handles small data and they can't ignore SIMD overhead.

> math operations

Ruby uses GMP if exist.

> sum, min, max, arrays, matrixes

Normal array can store any type.
To use SIMD power, the array should be typed array like NArray.
It's not Ruby itself's issue.

> string manipulations

I tried to use SSE2 for coderange_scan() in string.c, but it doesn't improve performance so much.

SSE 4.2 STTNI is also interesting but I don't find a good use case which can pay for increasing code complexity.

#2 - 07/06/2018 01:57 PM - nobu (Nobuyoshi Nakada)

- Status changed from Open to Closed

#3 - 07/16/2018 05:17 AM - nobu (Nobuyoshi Nakada)

- Status changed from Closed to Open

#4 - 07/16/2018 05:12 PM - ahorek (Pavel Rosický)

naruse (Yui NARUSE) I saw your blank implementation, impressive
https://github.com/ruby/ruby/commit/e6bc209abf81d53c2e3374dc52c2a128570c6055

the complexity for a hand written simd code is probably too high. Ruby supports a lot of platforms, so we have to duplicate the code (compatibility paths) or make a portable interface for it.

here's also an interesting implementation of "strip" method
https://github.com/lemire/despacer

I don't like the idea of exposing simd types like NArray to the developer, but some languages did it this way (like Dart)

The best solution is to teach JIT how to vertorize at least basic loops like

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

->

for (int i = 0; i < N/8; ++i)
  VECTOR_ADD(A + i, B + i, C + i);
```

unfortunatelly it's not always as simple as this example

#5 - 01/08/2020 04:14 PM - naruse (Yui NARUSE)

- Related to Misc #16487: Potential for SIMD usage in ruby-core added