Orchestrating a brighter world NEC

Hybrid Accelerator
NEC SX-Aurora TSUBASA

2022年1月20日（木）

日本電気株式会社　政岡 靖久 <ymsk-bx@nec.com>

Agenda

SX-Aurora TSUBASAとは
使い方（VE Offloading）
使い方（Transparent Execution）
Roadmap
For Your Future

SX-Aurora TSUBASAとは

使い方（VE Offloading）

使い方（Transparent Execution）

Roadmap

For Your Future

History of SX Vector Supercomputer

SX-2 1983

Technology:	Bipolar
CPU Frequency:	166 MHz
CPU Performance:	1.3 GFlops
CPU Memory Bandwidth:	10.7 GB/sec

SX-3 1989

Technology:	Bipolar
CPU Frequency:	340 MHz
CPU Performance:	5.5 GFlops
CPU Memory Bandwidth:	12.8 GB/sec

SX-4 1994

Technology:	350 nm
CPU Frequency:	125 MHz
CPU Performance:	2.0 GFlops
CPU Memory Bandwidth:	16.0 GB/sec

SX-5 1998

Technology:	250 nm
CPU Frequency:	250 MHz
CPU Performance:	8.0 GFlops
CPU Memory Bandwidth:	64.0 GB/sec

SX-6 2001

Technology:	150 nm
CPU Frequency:	500 MHz
CPU Performance:	8.0 GFlops
CPU Memory Bandwidth:	32.0 GB/sec

SX-7 2002

Technology:	150 nm
CPU Frequency:	552 MHz
CPU Performance:	8.8 GFlops
CPU Memory Bandwidth:	35.3 GB/sec

SX-8 2004

Technology:	90 nm
CPU Frequency:	1.0 GHz
CPU Performance:	16.0 GFlops
CPU Memory Bandwidth:	64.0 GB/sec

SX-9 2007

Technology:	65 nm
CPU Frequency:	3.2 GHz
CPU Performance:	102.4 GFlops
CPU Memory Bandwidth:	256.0 GB/sec

SX-ACE® 2013

Technology:	28 nm
CPU Frequency:	1.0 GHz
CPU Performance:	256.0 GFlops
CPU Memory Bandwidth:	256.0 GB/sec

Over 35 years experience for High Sustained Performance

Features of SX-Aurora TSUBASA

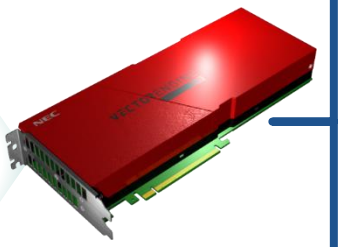
SX-ACE

(Traditional Super Computer)

Downsize!

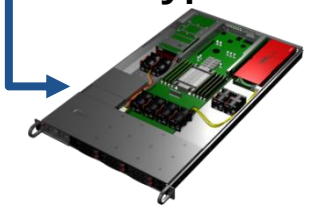
SX-Aurora TSUBASA

Vector Engine



Tower Type

Rack Type



Downsizing of super computer realized by NEC's Technology.

POINT 1 High Memory Bandwidth

Vector technology makes it possible to process multiple and huge data at a time with high memory bandwidth.

POINT 2 Ease of Use

No specialized knowledge is required, AP can be executed only after compiled.
Use C/C++/Fortran/Python to program.

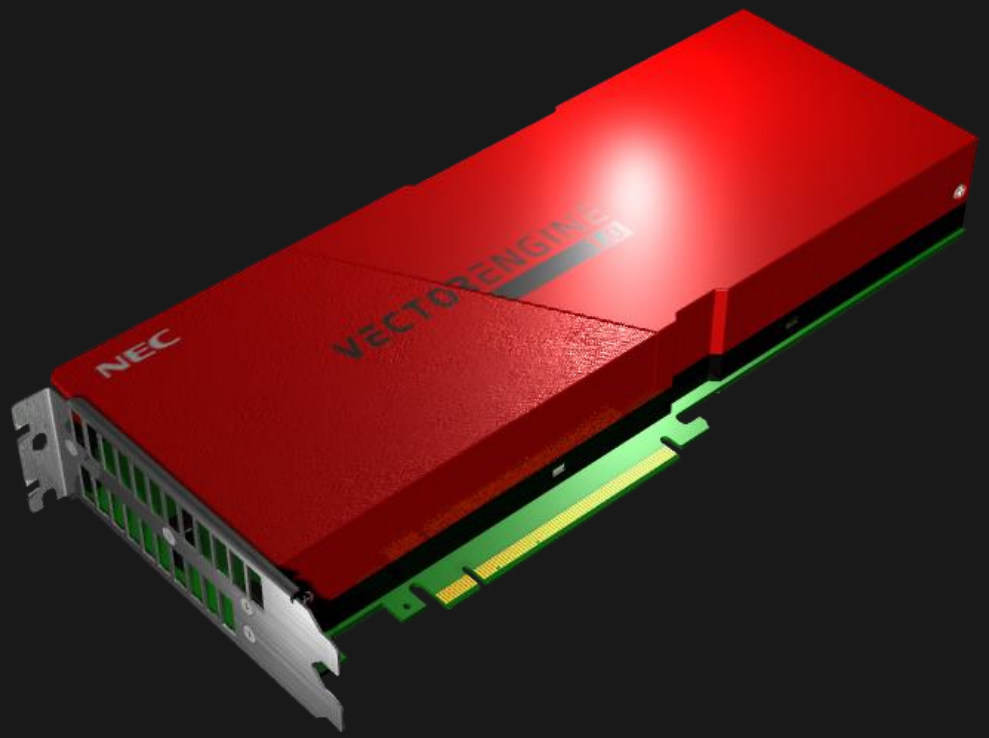
POINT 3 Flexibility

Customer can choose a system which meets their needs. From server type to card specification are all optional, NEC help customer to maximize the cost performance, to fit all market requirement.

Vector Engine

POINT 1 High Memory Bandwidth

◆ Vector technology is packed into a PCI card.



- Vector processor (8/10 cores)
- 1.53TB/s memory bandwidth
- 48GB memory
- 2.45/3.07TF performance (double precision)
  4.91/6.14TF performance (single precision)
- A variety of execution modes
- Standard programming with C/C++/Fortran/Python
- Power consumption < 300 W

Lineup of SX-Aurora TSUBASA

POINT 3 Flexibility

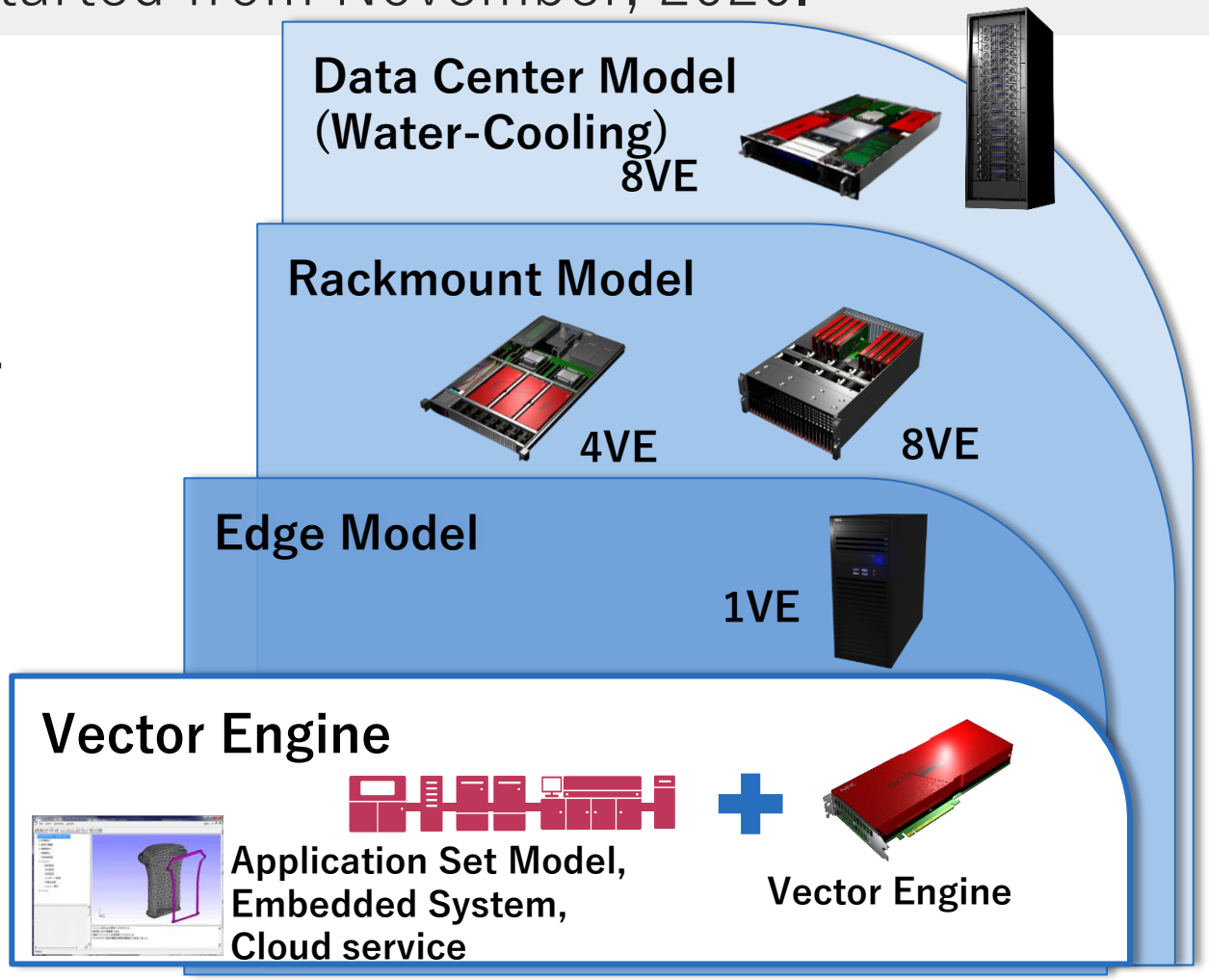
Vector Engine supports wide range from desk-side to large-scale Data Centers.
Selling Vector Engine card was started from November, 2020.

Data Center Model
Huge processing in Data Centers

Rackmount Model
Simulation of manufacturing industry, etc.
Use of AI / big data

Edge Model
Simulation of mid-sized manufacturing industry, etc. Laboratory desk-side

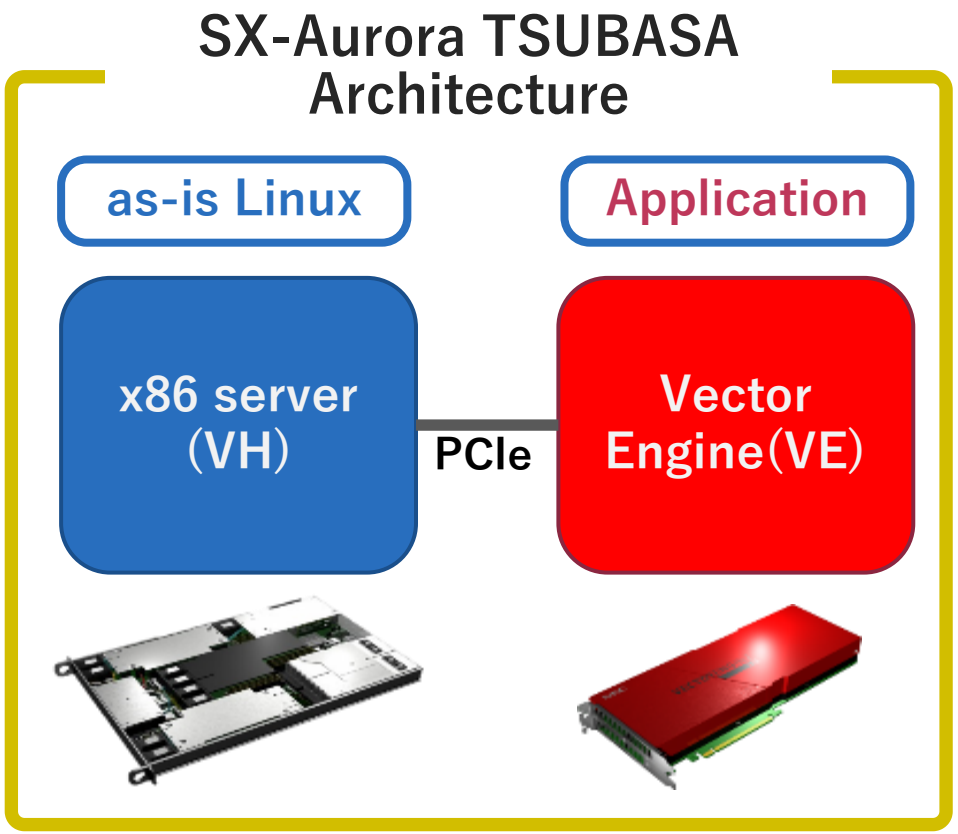
Vector Engine
Application acceleration
Embedded use
Cloud service engine



Architecture of SX-Aurora TSUBASA

POINT 2 Ease of Use

- SX-Aurora TSUBASA = VH + VE
- Linux + standard language (C/C++/Fortran/Python)
- Enjoy high performance with easy programming



Hardware

- VH(Standard x86 server) + Vector Engine

Software

- as-is Linux
- C/C++/Fortran/Python
- Automatic vectorization compiler

Interconnect

- InfiniBand for MPI
  - ✓ VE-VE direct communication support

Easy programming (standard language) ▶ Automatic vectorization compiler ▶ Enjoy high Performance!

SX-Aurora TSUBASAとは

使い方（VE Offloading）

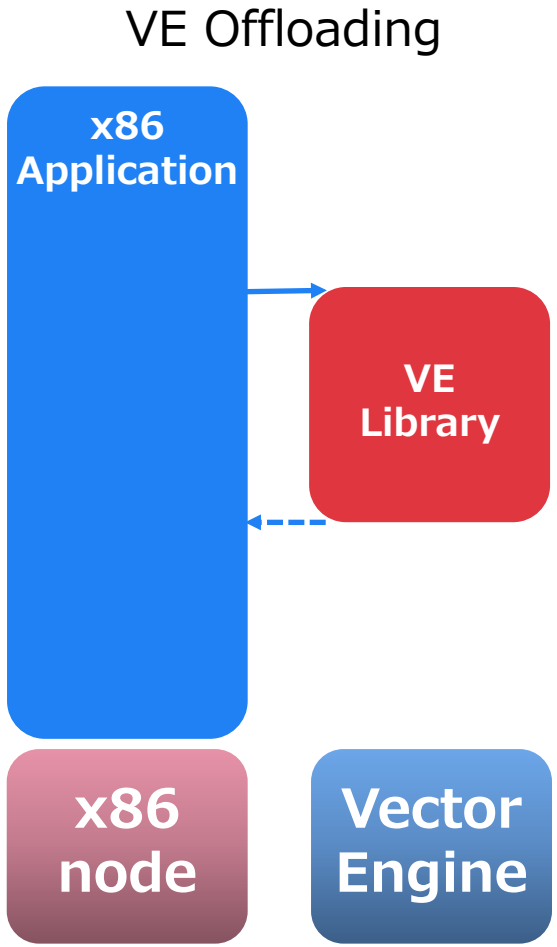
使い方（Transparent Execution）

Roadmap

For Your Future

SX-Aurora TSUBASA programming model

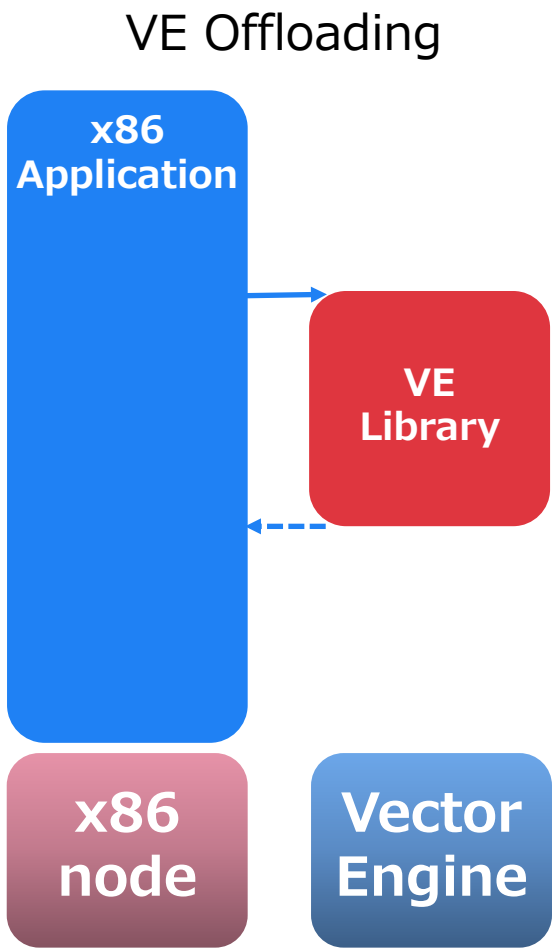
VE Offloading



- VEをアクセラレータ方式で使うためのフレームワーク
- プログラムの本体はx86（VH）側にあり、その一部をVE側で必要に応じて実行できる。

How to use VE Offloading?

VEライブラリはC/C++、Fortranで記述可能。



- x86 Application（プログラム本体）
  - C/C++用のAPIをサポート
- VE Library（プログラムの一部）
  - vi libve.c

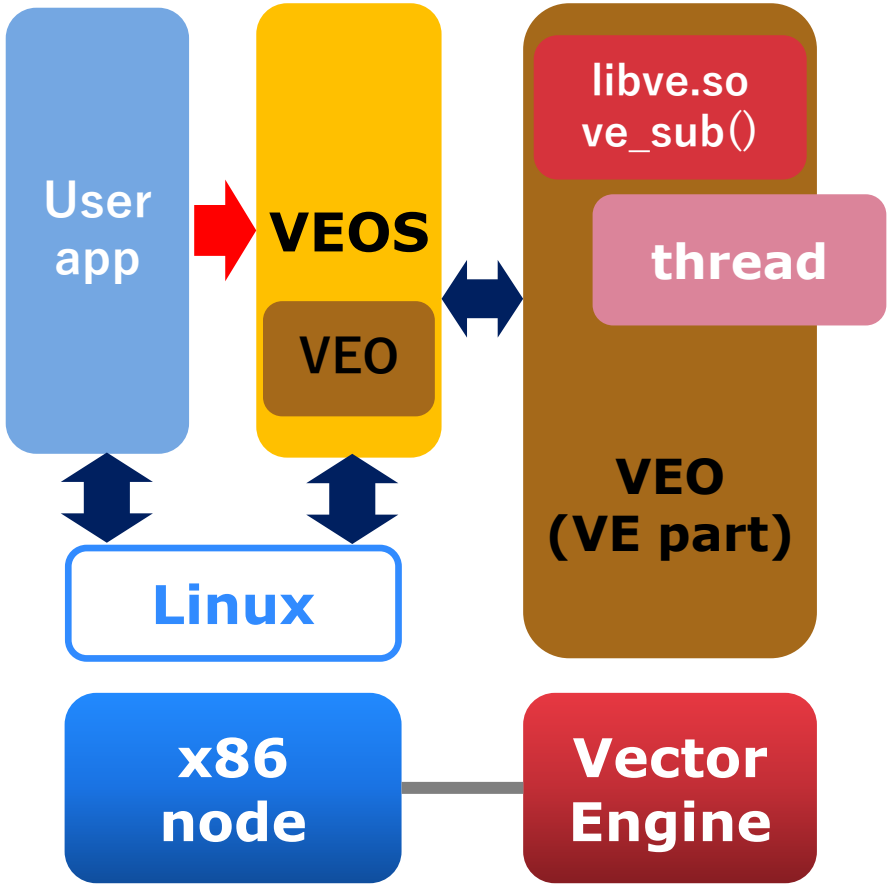
```
uint64_t ve_data[64*1024*1024/sizeof(uint64_t)];
int ve_sub(int arg1, ...)
{
  for (uint64_t i = 0; i < ...) {
    ve_data[i] += ...;
  }
  return(0);
}
```

  - /opt/nec/ve/bin/ncc libve.c -shared –fpic -o libve.so

（つづき）How to use VE Offloading?

APIs for x86 (VH) application to offload a part of program to VE.

libveo API for C/C++



1.Create process on VE

```
veo_proc_handle *veo_proc_create( int venode)
```

2.Load shared library on VE

```
uint64_t veo_load_shared_library( veo_proc_handle *proc,  const char *libname)
```

3.Create thread on VE

```
veo_thr_ctx *veo_context_open( veo_proc_handle *proc)
```

4.Create arguments of func()

```
veo_args *veo_args_alloc(void)
```

```
int veo_args_set_<type>(veo_args *ca, int argnum, <type> val)
```

5.Call func() asynchronously by name

```
uint64_t veo_call_async_by_name( veo_thr_ctx *ctx, uint64_t libhdl,
                                 const char *symname, veo_args *args)
```

6.Wait result of func()

```
int veo_call_wait_result( veo_thr_ctx *ctx, uint64_t reqid, uint64_t *retp)
```

NLCPy

PythonスクリプトからNLCの一部機能をご利用いただけます
NumPyスクリプトはモジュール名を変更するだけでVEの演算能力をご活用いただけます

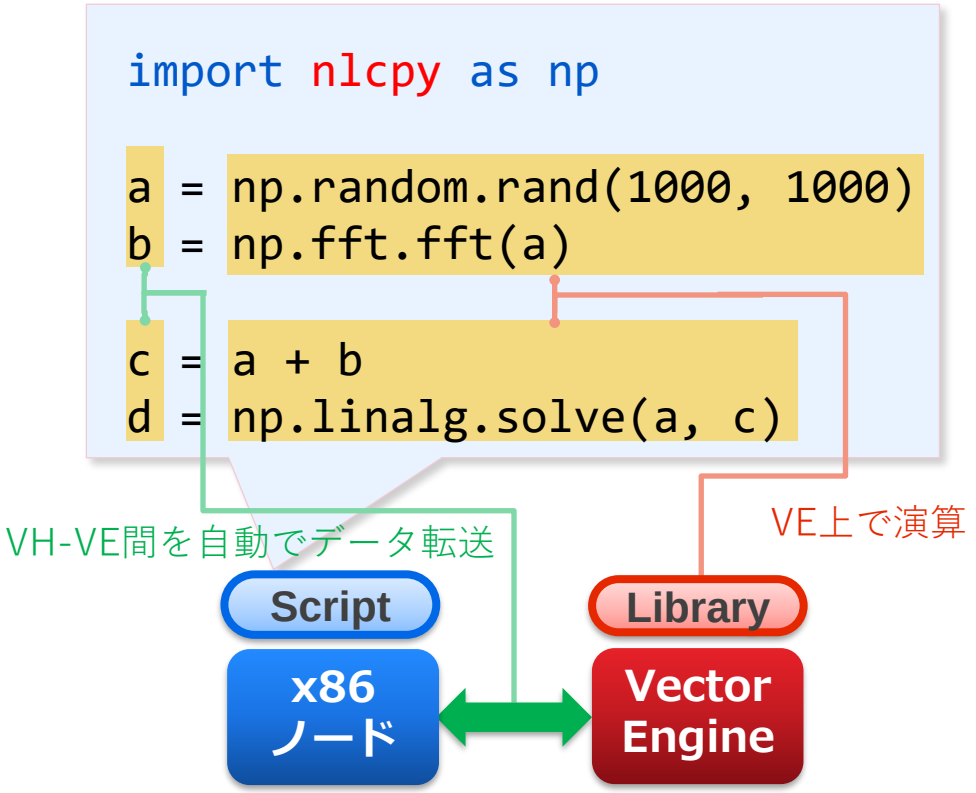
◆NLCPyの特長

■NumPyからの移植が容易

- NumPyのAPIのサブセットを提供

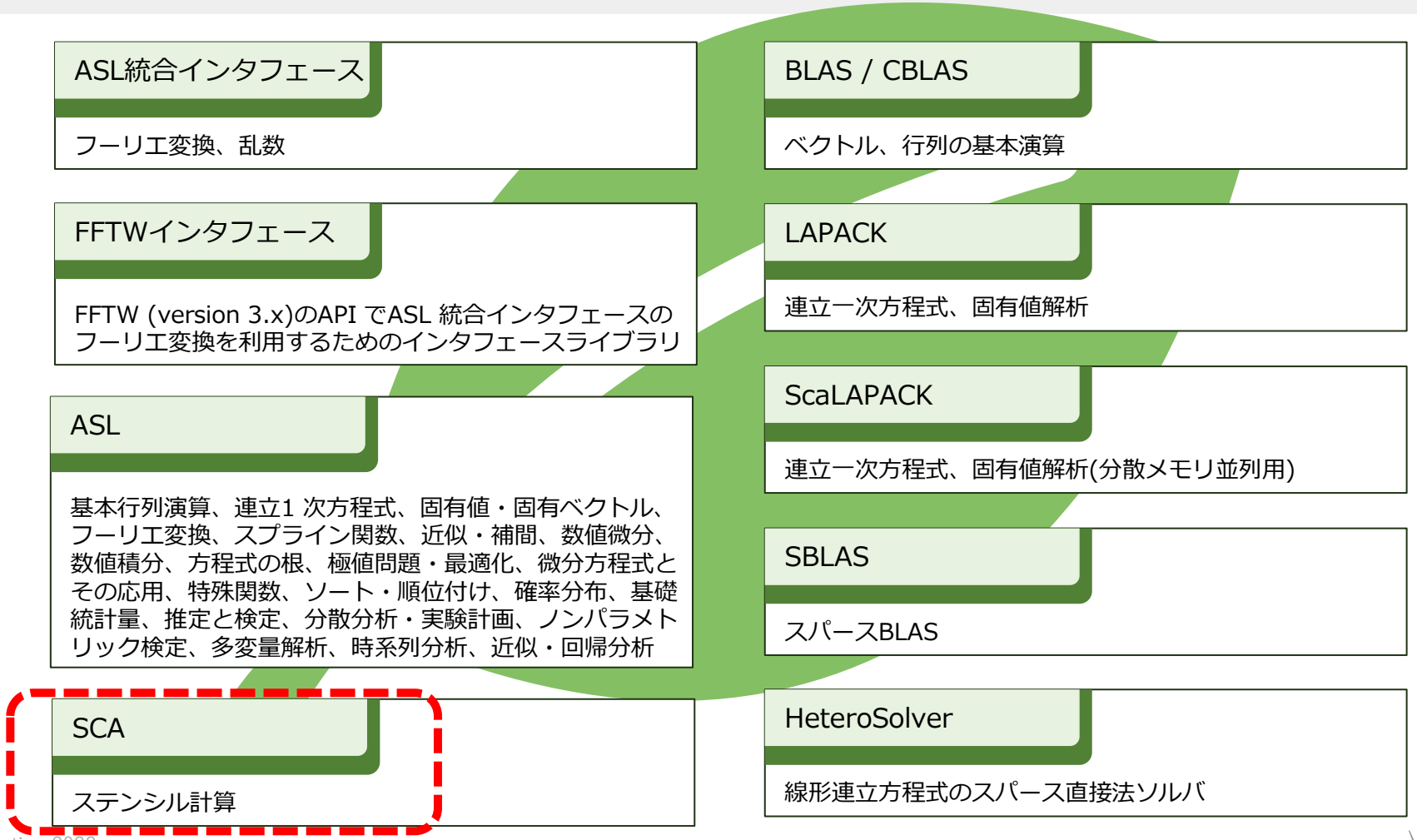
■VE向けに最適化

- VE向けに最適化されたライブラリを利用
  BLAS、LAPACK、 ASL、SCA等
- ベクトル化された演算機能を多数提供
  四則演算、数学関数、乱数、ソート、FFT等
- VE上で実行する関数を非同期に呼び出し、
  VH-VE間オフロードのオーバヘッドを削減



NEC Numeric Library Collection (NLC)

NLCは、広範な分野の数値シミュレーションプログラムの作成を強力に支援する数学ライブラリのコレクションであり、Vector Engineに対応しています



Stencil calculation Code Accelerator Interface of NLCPy

- SCA Interface
  - Boosts performance of a wide variety of stencil calculations.
- Example
  - User can easily define arbitrary stencil shapes by specifying relative locations.

```
import nlcpy as vp

a = vp.random.rand(100).reshape(10,10)
b = vp.zeros_like(a)

din, dout = vp.sca.create_descriptor((a, b))
desc_in  = 0.25 * ( din[0,-1] + din[-1,0] + din[1,0] + din[0,1] )
desc_out = dout[0,0]
kern = vp.sca.create_kernel(desc_in, desc_o=desc_out)

for i in range(maxitr):
    a[...] = kern.execute()
```

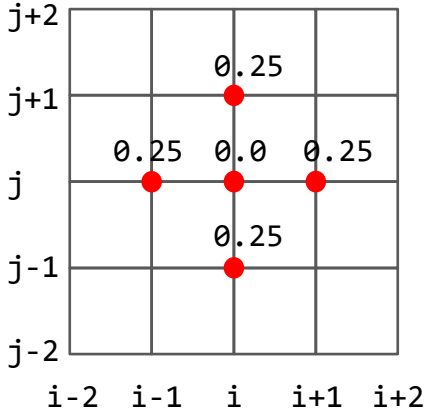
Laplace Equation

$$\frac{\partial}{\partial x^2}A + \frac{\partial}{\partial y^2}A = 0$$

discretization

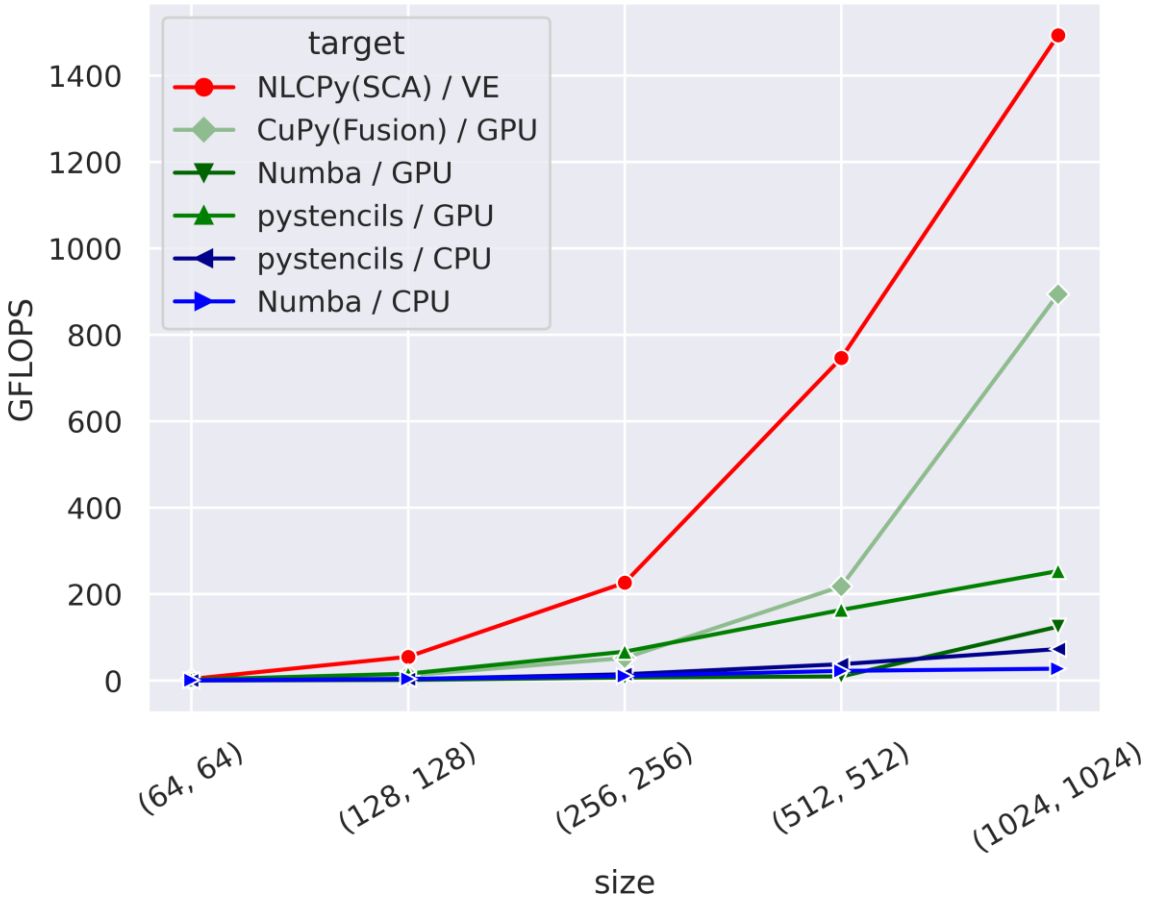
finite-difference

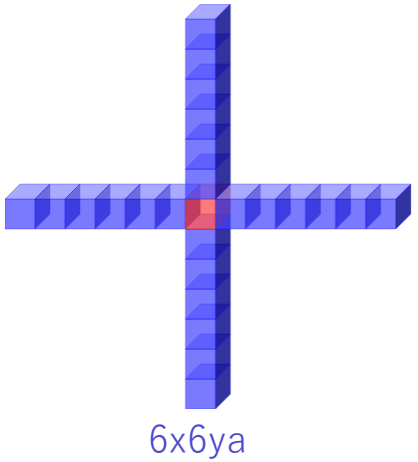
$$b_{i,j} = \frac{1}{4}\left(a_{i,j-1} + a_{i-1,j} + a_{i+1,j} + a_{i,j+1}\right)$$



Benchmark Result (2-D XY-axial stencils)

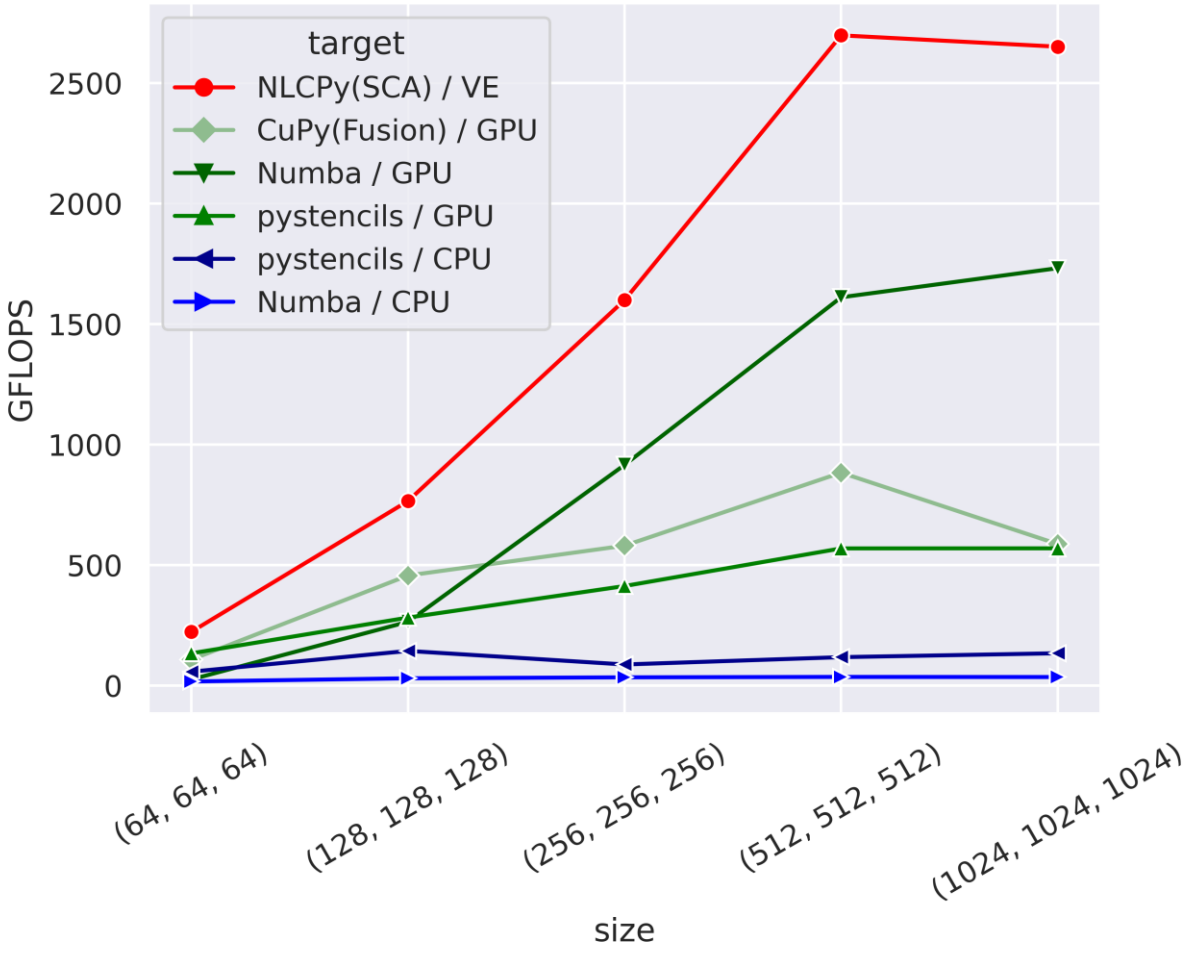
- NLCPy on Vector Engine shows the highest performance.

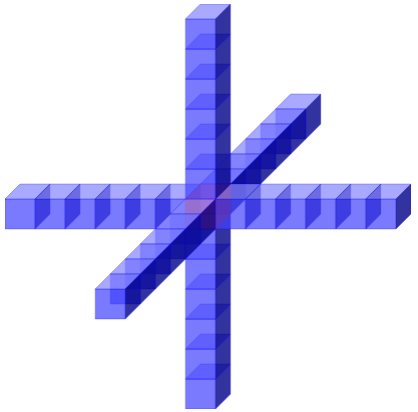




Benchmark Result (3-D XYZ-axial stencils)

- NLCPy on Vector Engine shows the highest performance.

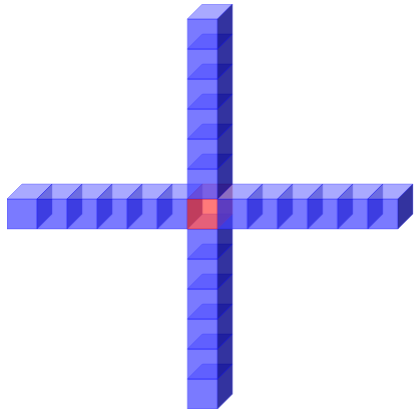




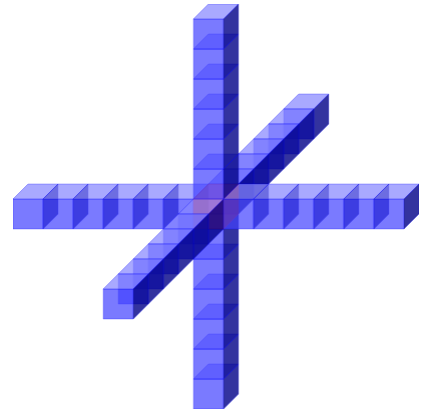
6x6y6za

Benchmark Conditions

- Stencil Shapes and Time Steps
  - 2-D: XY-axial, size 6.
  - 3-D: XYZ-axial, size 6.
  - For 100 time steps.



6x6ya



6x6y6za

- Target Libraries
  - Numba:
    - A100 PCI-E 40GB
    - Xeon Gold 6126 x2 (Skylake 2.60GHz, 48 cores)
  - pystencils:
    - A100 PCI-E 40GB
    - Xeon Gold 6126 x2 (Skylake 2.60GHz, 48 cores)
  - CuPy on A100 PCI-E 40GB
  - NLCPy on VE Type 20B (8 cores)

- Data Size
  - 2-D: (64, 64) (128, 128) (256, 256) (512, 512) (1024, 1024)
  - 3-D: (64, 64, 64) (128, 128, 128) (256, 256, 256) (512, 512, 512) (1024, 1024, 1024)

  For each case, single precision was used.

NLCPy : Just-In-Time Compilation for VE

NLCPy provides Just-In-Time(JIT) compilation functionality.
By using this, you can customize your C/C++/Fortran kernel directly on your Python program.

◆ Example of JIT Compilation

```
import nlcpy

c_source = r"""
void ve_add(double *px, double *py, double *pz, int n) {
    for (int i = 0; i < n; i++) pz[i] = px[i] + py[i];
}
"""

lib = nlcpy.jit.CustomVELibrary(code=c_source)

ve_add = lib.get_function(
    've_add',
    args_type=('uint64_t', 'uint64_t', 'uint64_t', 'int32_t'),
    ret_type='void',
)

x = nlcpy.random.rand(10)
y = nlcpy.random.rand(10)
z = nlcpy.empty(10)

ve_add(x.ve_adr, y.ve_adr, z.ve_adr, z.size)
```

Define C source

Build & Load

Find a symbol of a function

Execution

SX-Aurora TSUBASAとは

使い方（VE Offloading）

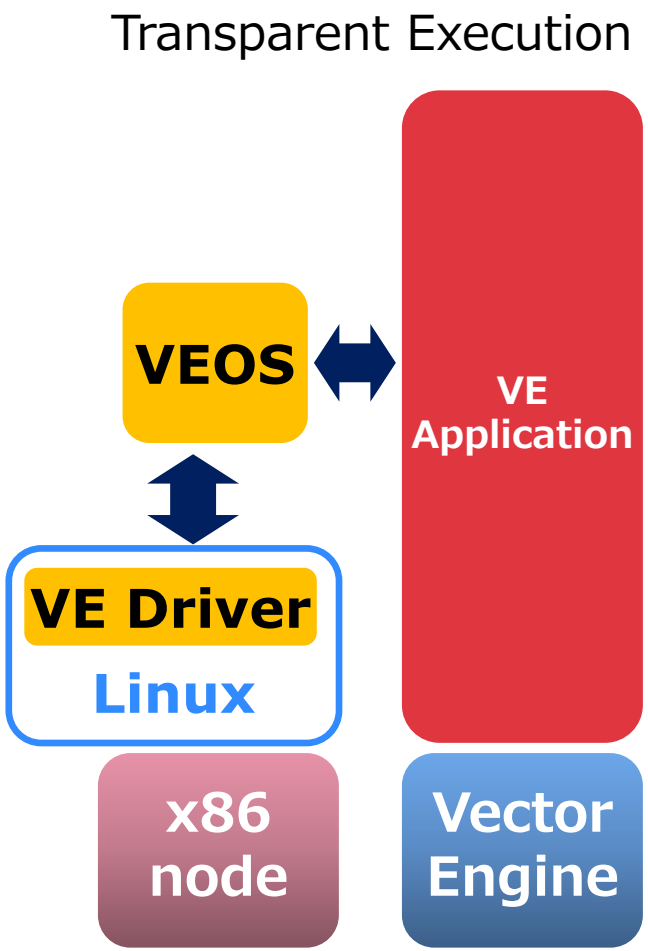
使い方（Transparent Execution）

Roadmap

For Your Future

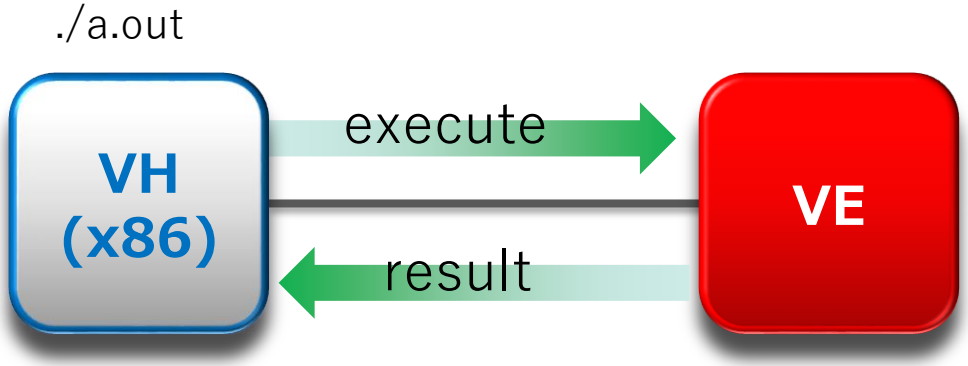
SX-Aurora TSUBASA programming model

Transparent Execution



- プログラムの実体はVE側に全てある
- 特別なプログラミングは必要なし
  - with **glibc**, C/C++, Fortran
- アクセラレータであることを意識せずに使用可能
  - **Hybrid Accelerator**
    - OS機能が必要な時はx86（VH）側に処理が自動的にオフロードされる
      - システムコール契機
    - x86（VH）側からのOS制御も可能
      - 実行開始/停止/終了、シグナル送信、 **gdbによるデバッグ**、コンテキストスイッチ、、他

Transparent Execution




```
$ vi sample.c
$ gcc sample.c
$ ./a.out
Hello World !
$
$ ncc sample.c
ncc: opt(1135): sample.c, line 10: Outer loop conditionally
executes inner loop.
ncc: vec( 101): sample.c, line 13: Vectorized loop.
$ ./a.out
Hello World !
```

Program on VH (x86)

Compile using ncc

Compiler message

Execute on VE

Result shown on VH

Trusted and Chose by World Famous HPC Centers

Český hydrometeorologický ústav : Weather Climate

Deutscher Wetterdienst
Wetter und Klima aus einer Hand : Weather / Climate

NIFS : Fusion Science
National Institute for Fusion Science

Tohoku university : Academic

JAMSTEC : Earth Science

Osaka university：Academic

Performance of large-scale computer system

JAMSTEC Earth Simulator is ranked in TOP10 in the latest HPCG ranking.
High Byte/Flops and high performance single core => High execution efficiency

JAMSTEC

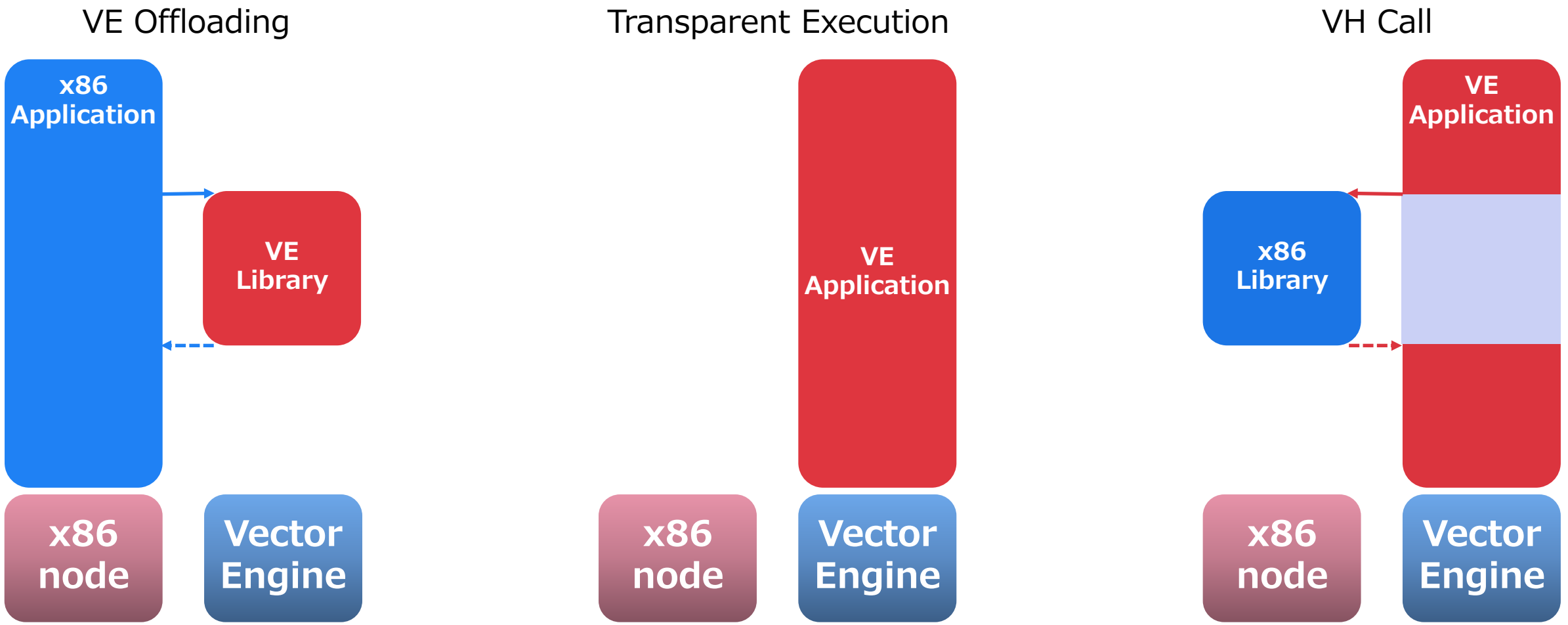
NIFS
National Institute for Fusion Science

Rank				Cores	HPCG [TFlop/s]	Rpeak [TFlop/s]	Execution efficiency
HPCG	HPL	System	Vendor				
1	1	Fugaku	Fujitsu	7,630,848	16,004.50	537,212.00	2.98%
2	2	Summit	IBM	2,414,592	2,925.75	200,794.88	1.46%
3	5	Perlmutter	HPE	706,304	1,905.44	89,794.48	2.12%
4	3	Sierra	IBM / NVIDIA / Mellanox	1,572,480	1,795.67	125,712.00	1.43%
5	6	Selene	Nvidia	555,520	1,622.51	79,215.00	2.05%
6	8	JUWELS Booster Module	Atos	449,280	1,275.36	70,980.00	1.80%
7	11	Dammam-7	HPE	672,520	881.40	55,423.56	1.59%
8	9	HPC5	Dell EMC	669,760	860.32	51,720.76	1.66%
9	13	Wisteria/BDEC-01	Fujitsu	368,640	817.58	25,952.26	3.15%
10	39	Earth Simulator - SX-Aurora TSUBASA	NEC	43,776	747.80	13,447.99	**5.56%**
11	25	TOKI-SORA	Fujitsu	276,480	614.22	19,464.20	3.16%
12	16	Trinity	Cray/HPE	979,072	546.12	41,461.15	1.32%
13	54	Plasma Simulator	NEC	34,560	529.16	10,510.66	**5.03%**
14	14	Marconi-100	IBM	347,776	498.43	29,354.00	1.70%
15	15	Piz Daint	Cray/HPE	387,872	496.98	27,154.30	1.83%

https://www.top500.org/lists/hpcg/2021/06/

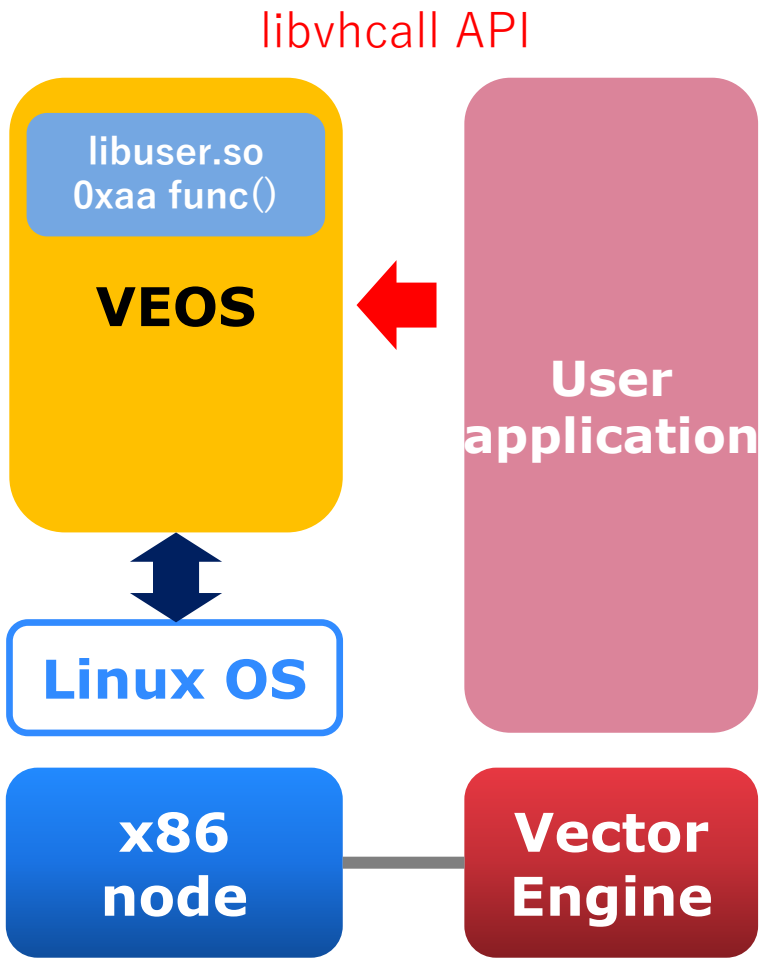
SX-Aurora TSUBASA programming model

The 3 models are available in Aurora architecture.



VH Call

Simple APIs for VE application to call library function on VH.



1.Load shared library on VH

```
vhcall_handle vhcall_install(const character *filename)
```

2.Find address of func() on VH

```
int64_t vhcall_find(vhcall_handle libhdl, const char *symname)
```

3 Create arguments of func()

```
vhcall_args *vhcall_args_alloc()
```

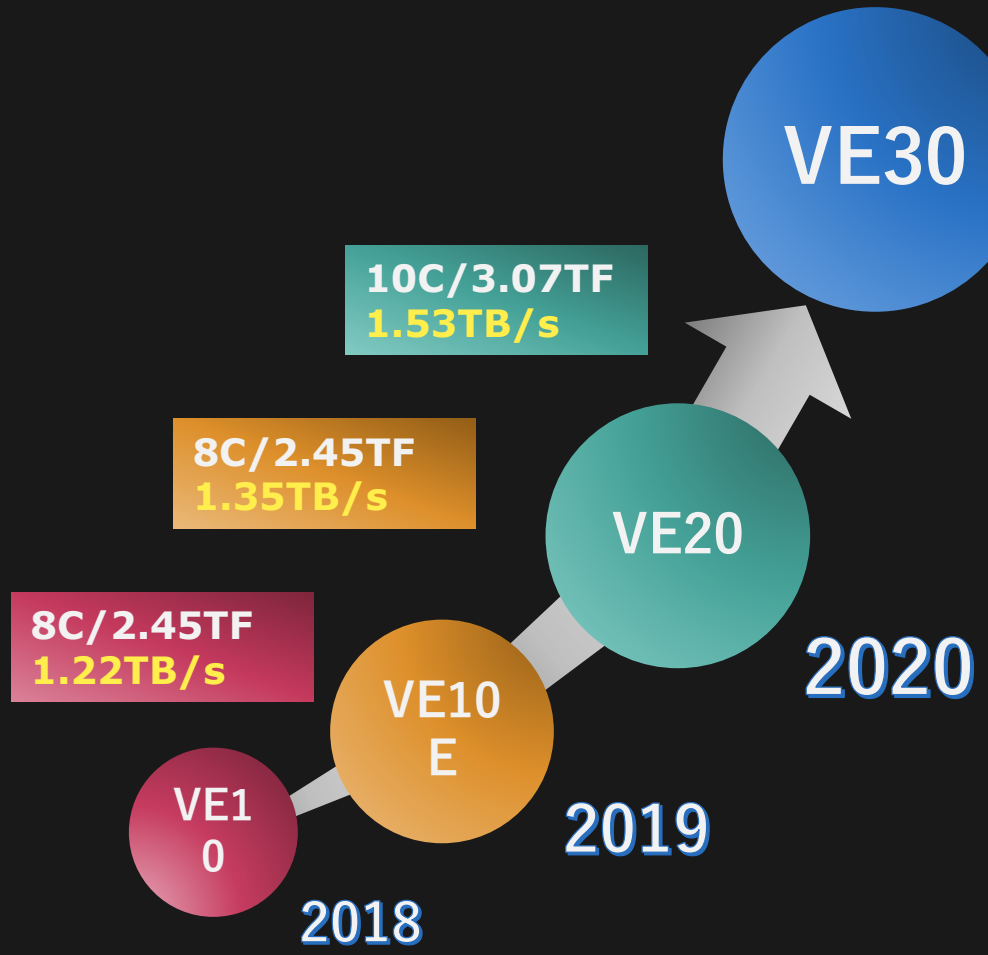
```
int vhcall_args_set_<type>(vhcall_args *args, int argnum, <type> val)
```

4.Call func() synchronously

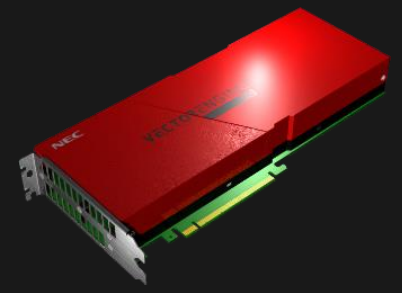
```
int vhcall_invoke_with_args(int64_t symid, vhcall_args *args, uint64_t *retptr)
```

SX-Aurora TSUBASAとは
使い方（VE Offloading）
使い方（Transparent Execution）
Roadmap
For Your Future

Vector Engine 3.0

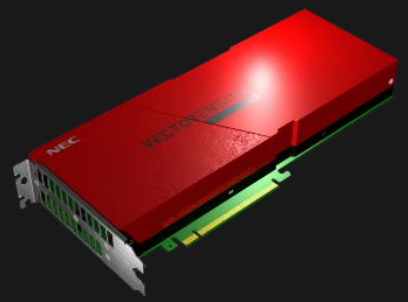


2+TB/s memory bandwidth



- **Targeting the largest memory bandwidth**
- Inheriting and improving VE/VH architecture
- Higher Flops per processor
- Improved memory subsystem including cache
- Accelerating short vector, and scalar operations
- Maintaining high power efficiency
- Virtual machine support

Roadmap



Memory bandwidth / VE

VE10

8C/2.45TF
1.22TB/s memory bandwidth

VE10E

8C/2.45TF
1.35TB/s memory bandwidth

VE20

10C/3.07TF
1.5TB/s memory bandwidth

VE30

2+TB/s memory bandwidth

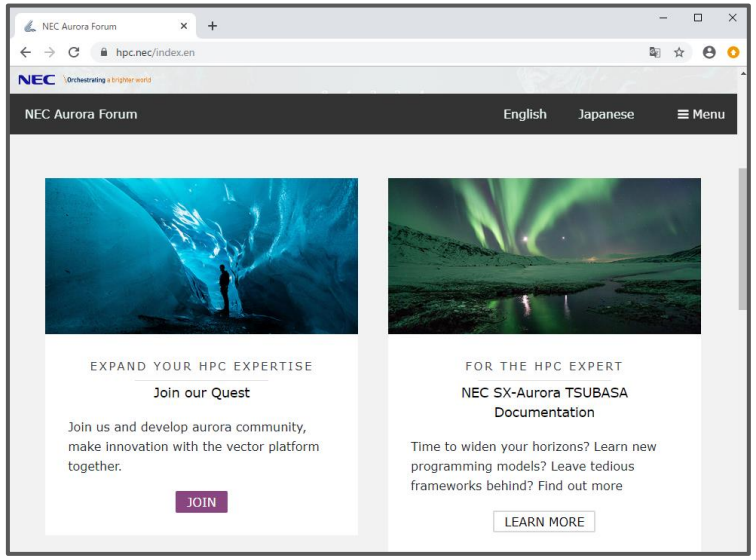
VE40

2++TB/s memory bandwidth

VE50

2019 2020 2021 2022 2023 2024

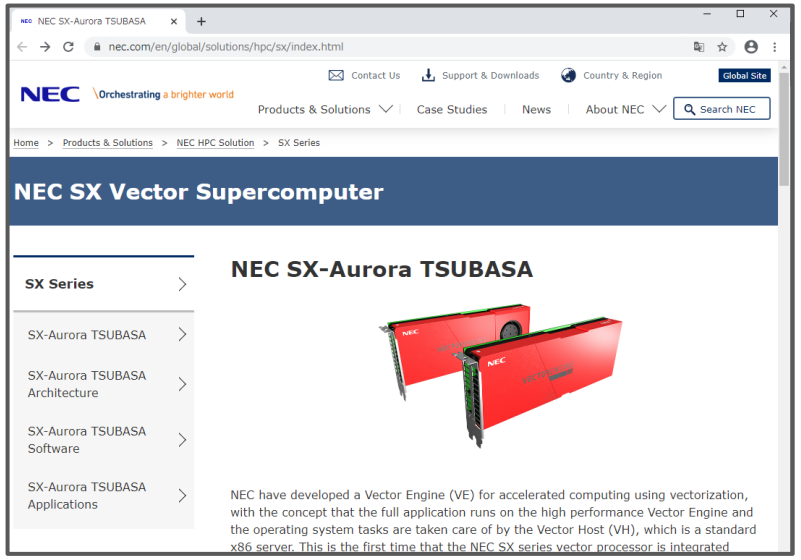
Find more information on our website



Aurora Web Forum

http://www.hpc.nec

- Latest updates
- Manual, documents
- Bulletin board



SX-Aurora TSUBASA Website

http://www.nec.com/en/global/solutions/hpc/sx/index.html

- Hardware and software overview
- Supported applications

✉ info@hpc.jp.nec.com

Tohoku university : Academic

JAMSTEC : Earth Science

Osaka university : Academic

SX-Aurora TSUBASA トライアルプログラム概要

- 目的
  - ご購入前評価を目的としたマシン環境の提供
- ご利用条件
  - 対象：SX-Aurora TSUBASAをご利用予定のお客様
    - 法人、大学、高等専門学校、他に所属されているお客様
  - 期間/回数：30日/1回限り
  - 費用： 無料（オンプレ貸出時の返送費用はお客様ご負担となります）
  - 形態： クラウド環境、ご要望に応じてオンプレ貸出
  - サポート：専用Webサイトでのマニュアル、チュートリアル情報、FAQのご提供
    専用フォームからのお問い合わせ対応
  - その他
    - 貸出期間中あるいは期間終了後に提供された質問、意見や提案、あるいは移植したOSSコードについて、他のお客様への公開を依頼させて頂く場合があります。
    - お客様からのフィードバックは、専用Webサイトなどで公開させて頂きます。
    - ご要望多数の場合は弊社で審査の上、貸出可否を判断させていただく可能性がございますのでご了承ください。

評価環境（一例）

SX-Aurora TSUBASA A300-2

CPU：Intel Xeon Gold 6126 (12core, 2.6GHz)　x1
MEM：96GiB
Disk：1TB HDD
VE：TypeC (8core, 1.4GHz, 24GB MEM) x1

info@hpc.jp.nec.com

SX-Aurora TSUBASAとは
使い方（VE Offloading）
使い方（Transparent Execution）
Roadmap
For Your Future

\Orchestrating a brighter world

NECは、安全・安心・公平・効率という社会価値を創造し、
誰もが人間性を十分に発揮できる持続可能な社会の実現を目指します。

Background of multi-architecture system -towards Heterogeneous Computing-

Architecture is selected according to characteristics of each of applications.
One of trends in HPC system is hybrid, composed of a variety types of processors.

Scientific calculation

- Weather forecast
- Aerodynamic analysis
- Collision analysis

Statistical processing

- Recommendation
- Demand prediction
- Fraud detection

Image recognition

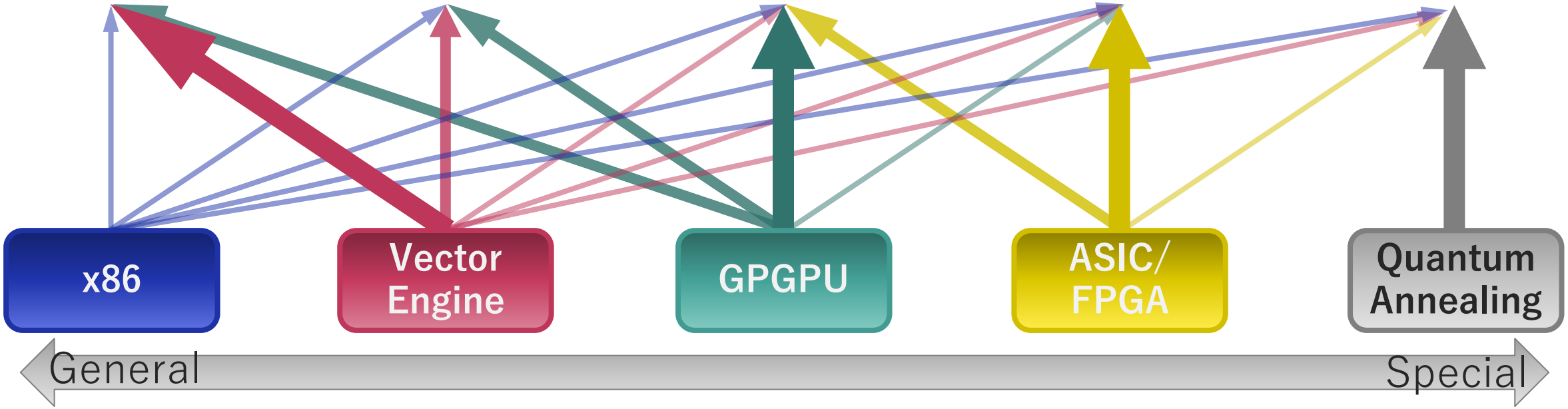
- Self-driving
- Checking goods
- Cancer diagnosis

Real-time transaction

- Financial transaction
- Face recognition
- Industrial robot

Combinatorial optimization

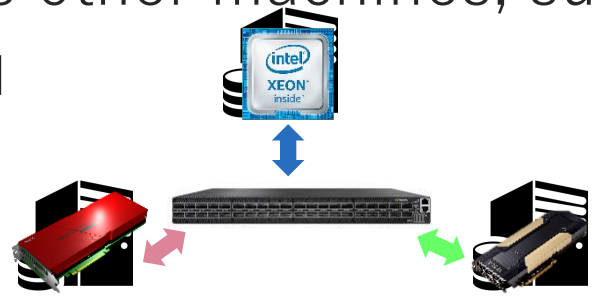
- Financial portfolio
- Shift schedule
- Delivery planning



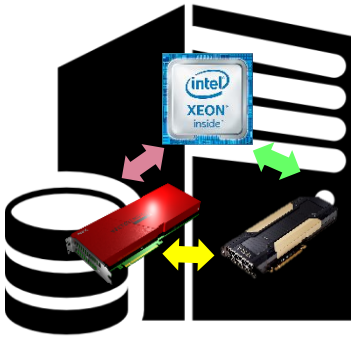
Aurora is ready for your future

- Aurora HPCG performance efficiency is better than the other machines.
  - The Fraction of Peak is over 5%. This will be one of the advantages of Aurora system in your future, especially when you want to build your LARGE computing system.

- Aurora can easily collaborate with the other machines, such as GPGPU, etc.
  - Through interconnect, right now Infiniband
    - Hybrid VE-x86/GPGPU MPI is available right now.



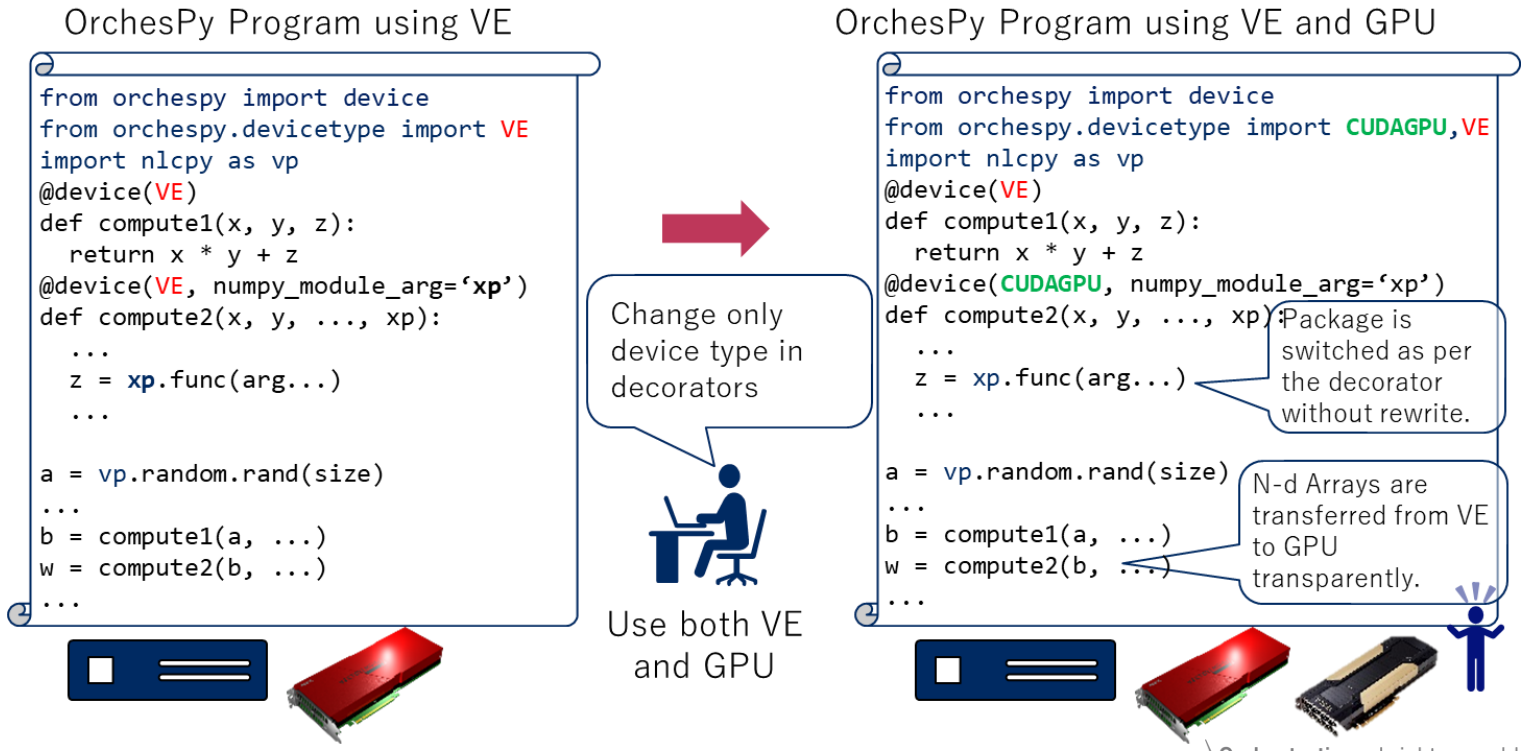
  - In VH (Linux/x86)
    - Not only VEs, but GPGPU, etc. can be installed in the same VH.
      - Hybrid VE-x86 MPI through PCIe is available right now.
      - Hybrid VE-GPGPU MPI through PCIe will be available in 2022.
      - NLCPy/Aurora with GPGPU will be available in 2022.



OrchesPy

OrchesPy is a NumPy-like library for VE-GPU heterogeneous systems.

- Feature
  - User will be able to choose execution devices just by specifying Python decorators.
- Schedule
  - April 2022 (Preview)



Tohoku university : Academic

JAMSTEC : Earth Science

Osaka university : Academic

SX-Aurora TSUBASA トライアルプログラム概要

- 目的
  - ご購入前評価を目的としたマシン環境の提供
- ご利用条件
  - 対象：SX-Aurora TSUBASAをご利用予定のお客様
    - 法人、大学、高等専門学校、他に所属されているお客様
  - 期間/回数：30日/1回限り
  - 費用： 無料（オンプレ貸出時の返送費用はお客様ご負担となります）
  - 形態： クラウド環境、ご要望に応じてオンプレ貸出
  - サポート：専用Webサイトでのマニュアル、チュートリアル情報、FAQのご提供
    専用フォームからのお問い合わせ対応
  - その他
    - 貸出期間中あるいは期間終了後に提供された質問、意見や提案、あるいは移植したOSSコードについて、他のお客様への公開を依頼させて頂く場合があります。
    - お客様からのフィードバックは、専用Webサイトなどで公開させて頂きます。
    - ご要望多数の場合は弊社で審査の上、貸出可否を判断させていただく可能性がございますのでご了承ください。

評価環境（一例）

SX-Aurora TSUBASA A300-2

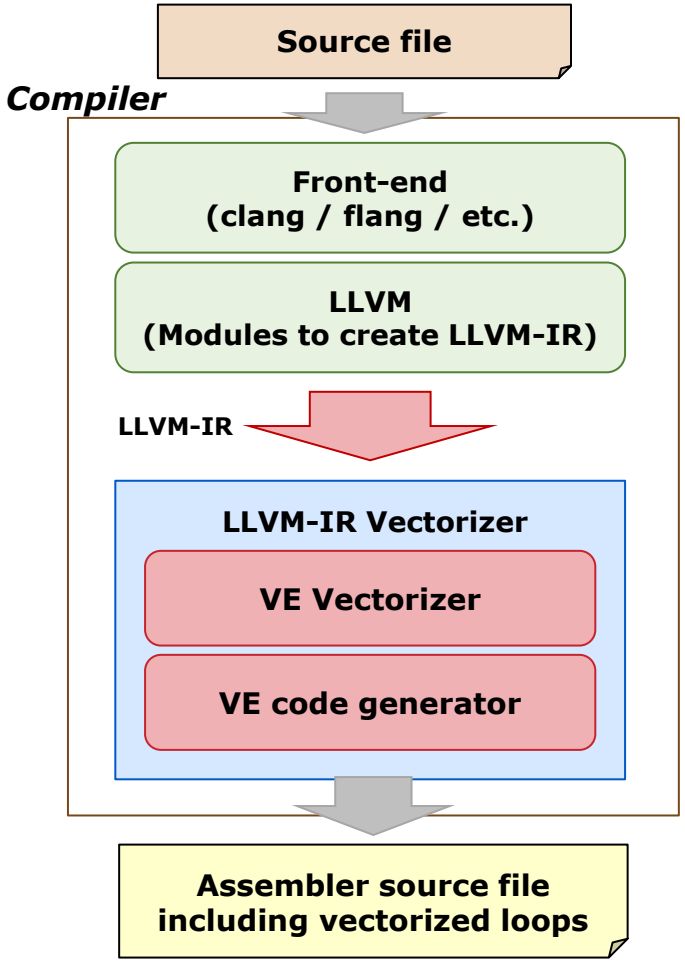
CPU：Intel Xeon Gold 6126 (12core, 2.6GHz)　x1
MEM：96GiB
Disk：1TB HDD
VE：TypeC (8core, 1.4GHz, 24GB MEM) x1

info@hpc.jp.nec.com

NEC LLVM-IR Vectorizer

https://www.hpc.nec/documents/llvm-vec/en/index.html

Provide binary plugin-module and runtime-library to add automatic vectorization feature for VE into your compilers



LLVM-IR Vectorizer

- 1st release based on LLVM 11.0 was May, 2021
  - Bug fix version was released at Aug, 2021 & Oct, 2021
- Includes vectorizer and code generator for VE.
- Inputs LLVM-IR from memory or an IR-file and outputs an assembler source code for VE.
- Applies automatic vectorization to LLVM-IR.
- Runtime library including vector mathematical functions (sin,cos etc.) and vector matrix-multiply functions is bundled.
- Source patch for clang to enable LLVM-IR Vectorizer is included. It helps to change your compiler's front-end

You can build your own compiler having vectorization feature for VE !!

VEOS and NLCPy

- VEOS概要設計
  - https://www.hpc.nec/documents/veos/jp/VEOS_high_level_design.pdf

- VEOS
  - https://github.com/veos-sxarr-NEC

- NLCPy
  - https://www.hpc.nec/documents/nlcpy/ja/index.html


Orchestrating a brighter world
NEC