Orchestrating a brighter world

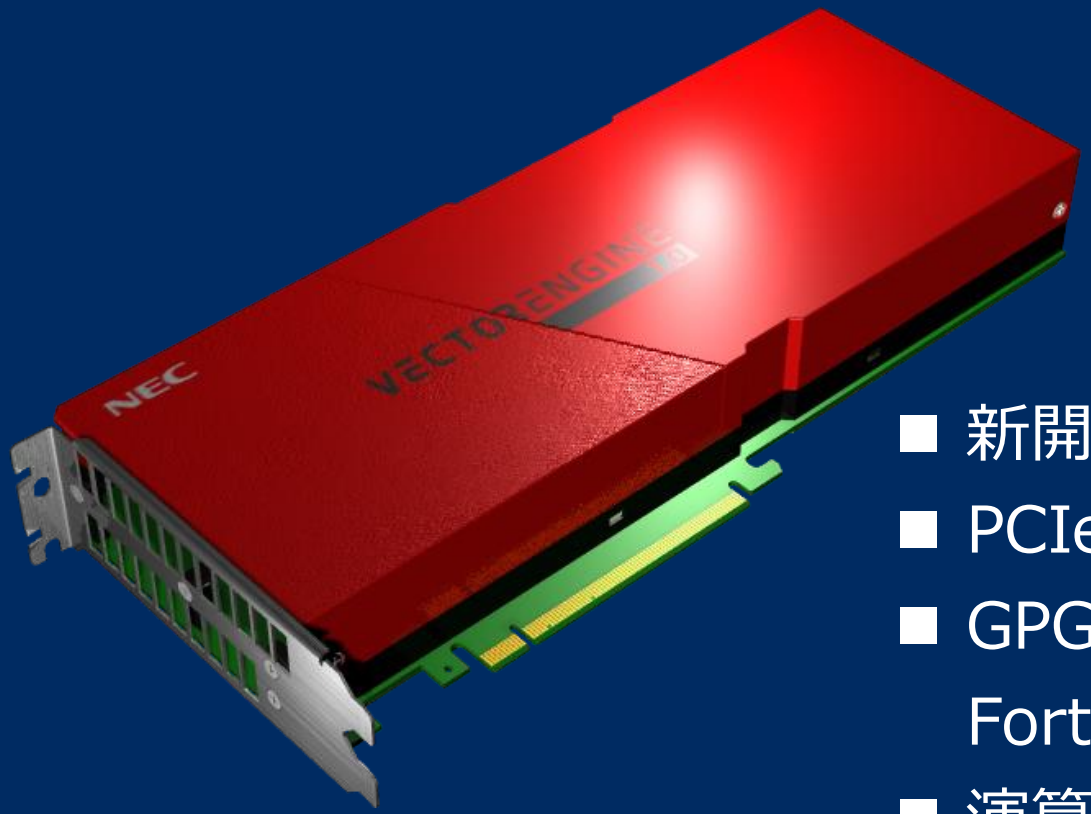
NEC

SX-Aurora TSUBASA　紹介資料

日本電気株式会社

2020年3月

PCIeカード型ベクトルエンジン

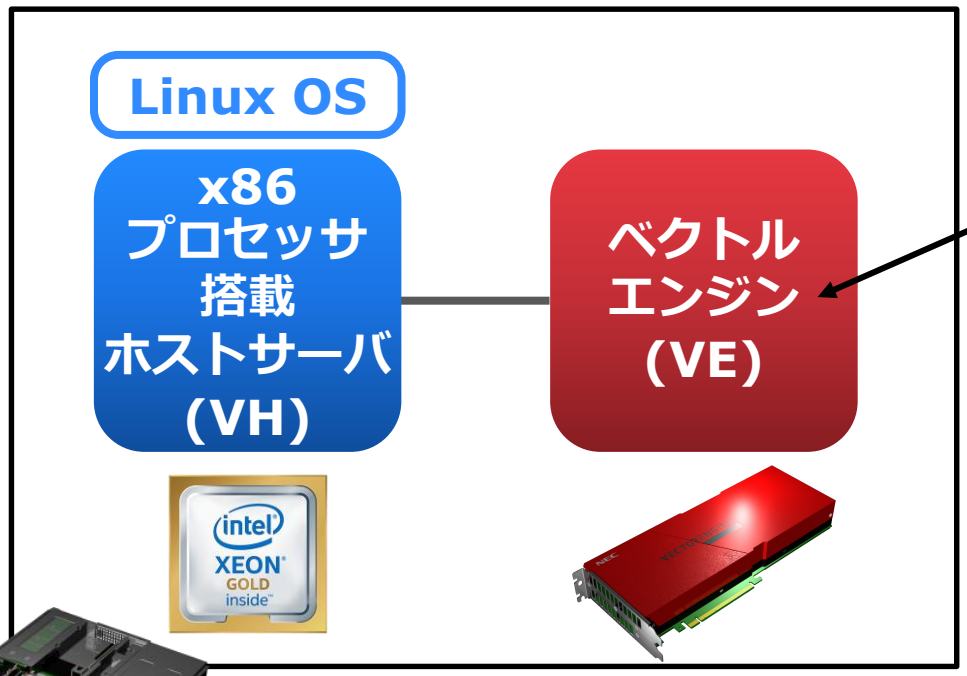


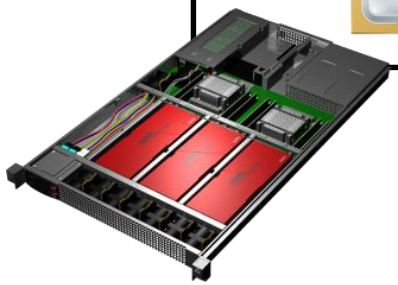
- 新開発ベクトルプロセッサ（8コア）
- PCIe規格準拠
- GPGPUと異なる実行モデル、Fortran/C/C++の標準環境
- 演算性能：
  2.43TF(倍精度)，4.87TF(単精度)
- メモリ帯域 1.35TB/s
  Xeonの1プロセッサの約10倍
- メモリ容量 48GB

Auroraのアーキテクチャ

特徴であるベクトルプロセッサをアクセラレータとして搭載

- ホストのX86プロセッサ
- ベクトルプロセッサを搭載したベクトルエンジン





Aurora

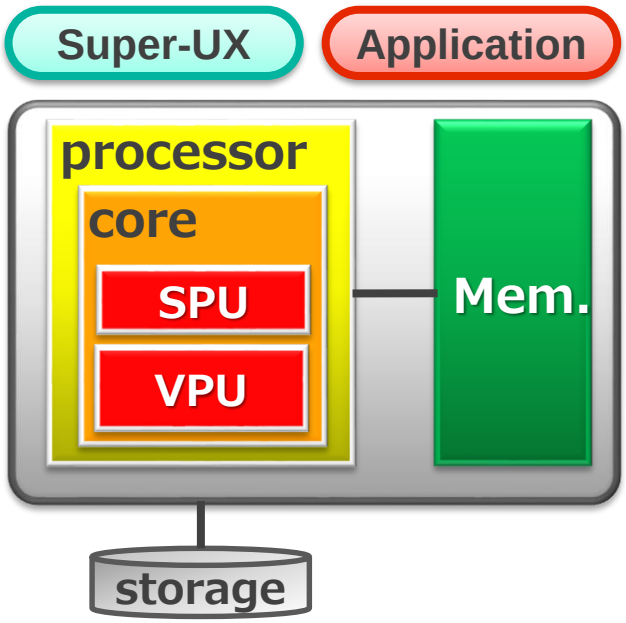
ベクトルプロセッサ

- **世界最速コア**
  - 304GFlops (DP)
  - 608GFlops (SP)
- **世界最速のデータアクセス性能**
  - 1.35TB/s
- **テクノロジ**
  - 世界初HBM2 x6実装

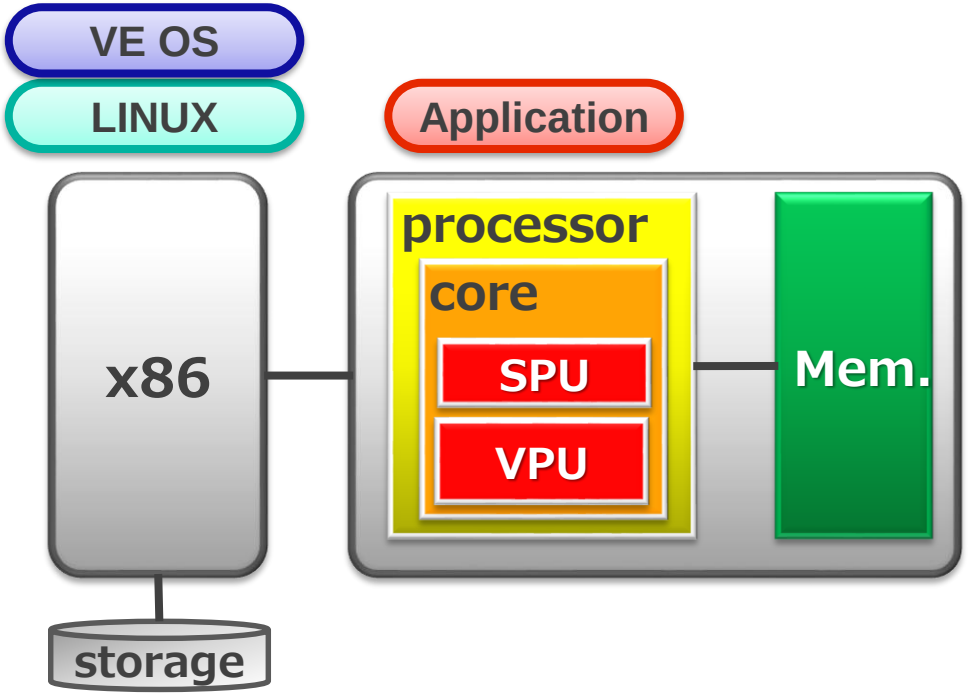
ベクトルエンジンの数は可変

従来SXシリーズとの違い

従来SX



Aurora



VH
Vector Host

VE
Vector Engine

SPU: Scalar Processing Unit
VPU: Vector Processing Unit

高性能：世界最速ベクトルプロセッサ

ベクトルプロセッサ

- **世界最速コア**
  - 304GFlops（DP）
  - 608GFlops（SP）

No.1

2019年10月現在、NEC調べ

- **世界最速のデータアクセス性能**
  - 1.35TB/s

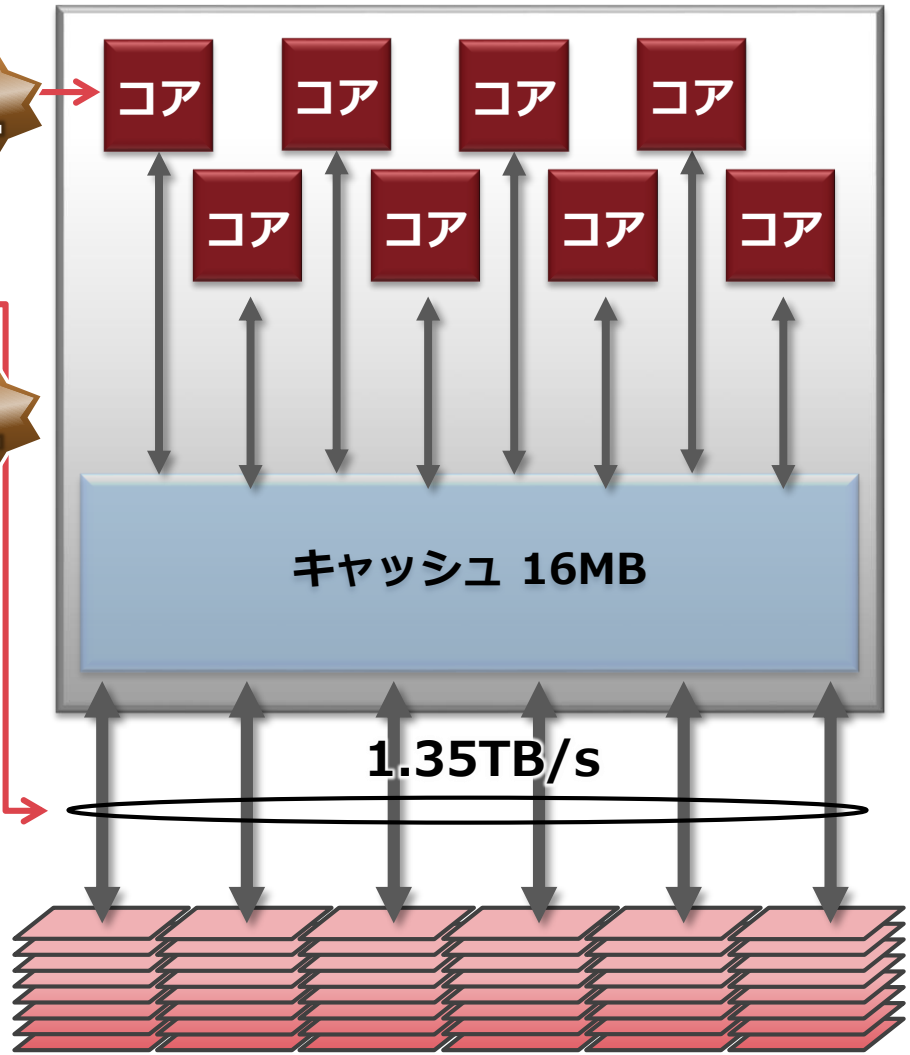
No.1

2019年10月現在、NEC調べ

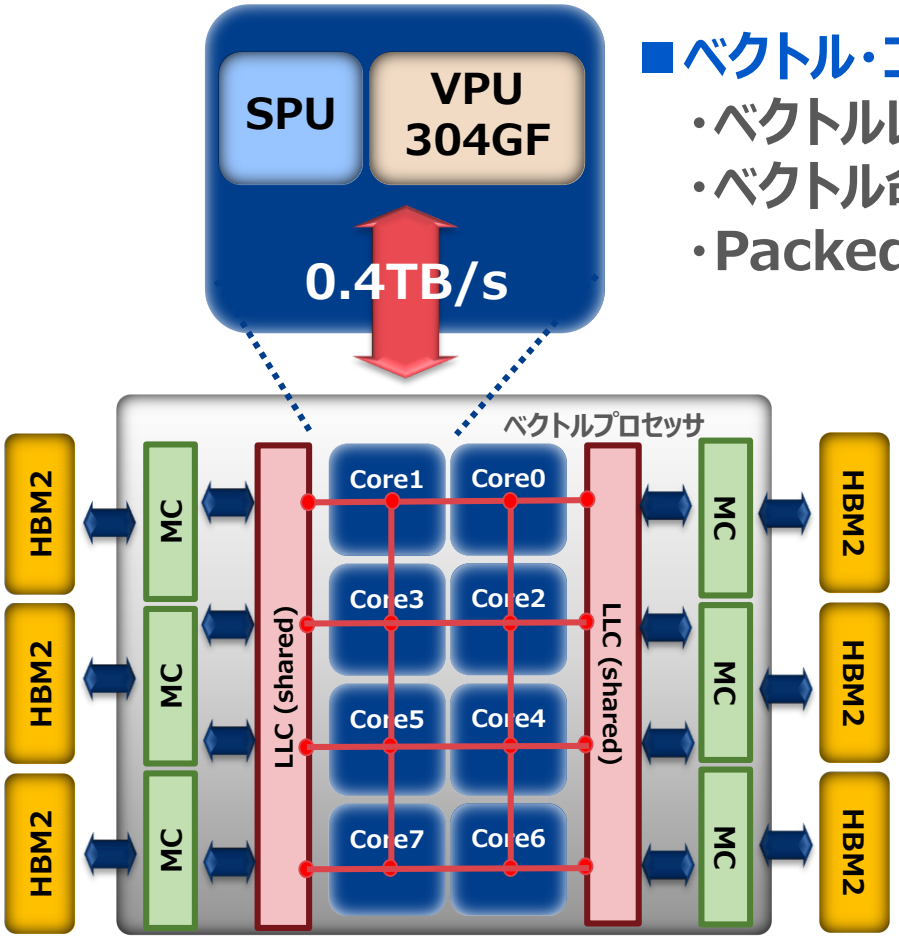
- **テクノロジ**
  - 世界初HBM2 x6実装

世界初

世界初となるCPUと6個の3次元積層メモリHBM2搭載技術をTSMC社と共同開発



ベクトルプロセッサ構成



■ベクトル・コア

- ベクトルレジスタのリネーミング機構
- ベクトル命令のOut of Order実行強化
- Packed 単精度(SP)演算の実装

■主な諸元

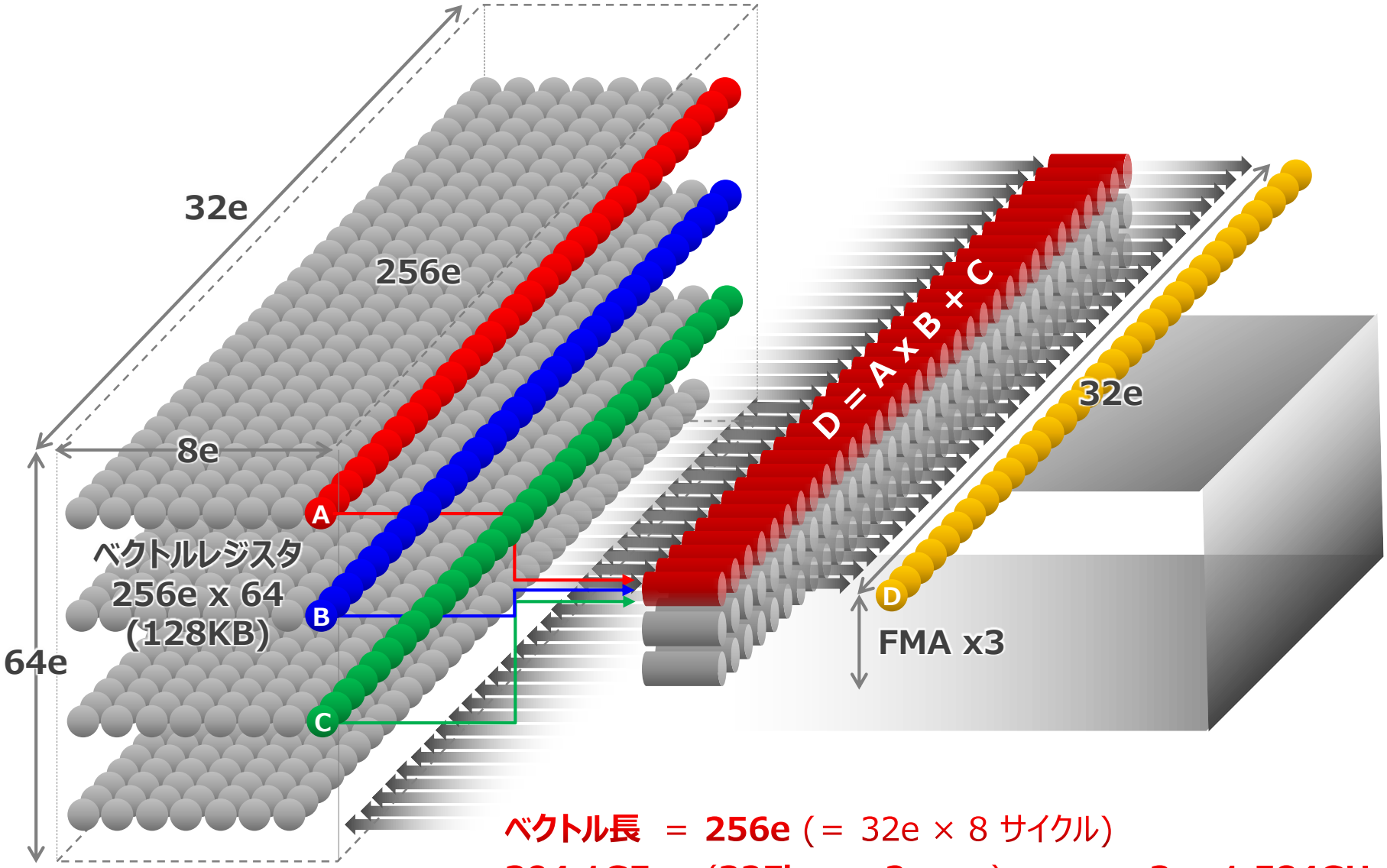
core数	8
core性能	~304GF(DP) ~608GF(SP)
CPU性能	~2.43TF(DP) ~4.86TF(SP)
cache 容量	16MB shared
メモリ バンド幅	0.75 ~ 1.35TB/s
メモリ容量	24 or 48GB

DP: Double Precision
SP: Single Precision

■ベクトルプロセッサ

- 2x4メッシュ・ネットワーク
- 16MBの共有キャッシュ

コア内のベクトル演算機構の演算器構成



ベクトル長 = 256e (= 32e × 8 サイクル)

304.1GF = (32Flop × 2 (FMA)) /サイクル **× 3 × 1.584GHz**

Aurora 1E: 製品ラインアップ

これまでのA1xx,A3xx,A5xxシリーズに高密度水冷タイプのA4xxシリーズを追加

Supercomputer Model
- For large scale configuration
- DLC with 40℃ water

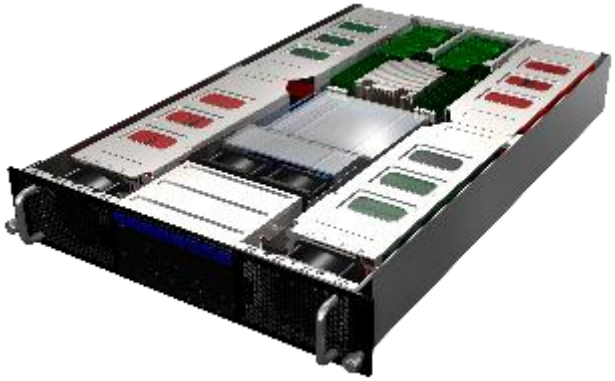
Water cool model
- 8VE in 2U High density
- DLC with 40℃ water

Rack Mount Model
- Flexible configuration
- Air Cooled

Tower Model
- For developer/programmer
- Office room use

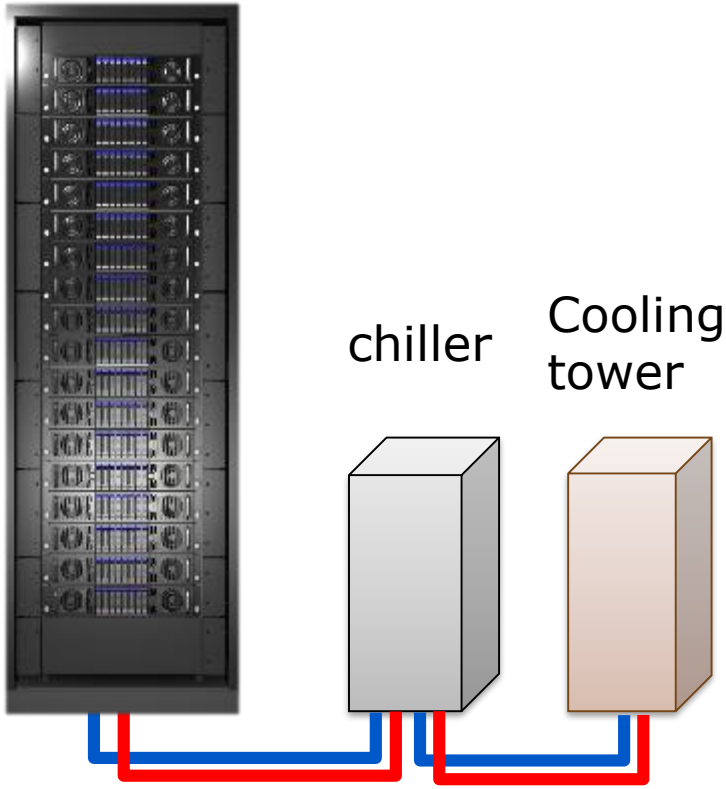
Aurora 1E：高密度 DLC サーバ

- A5xxシリーズ（64VE）2倍以上の実装密度
- 高温水（40℃）冷却で省スペース、省エネを実現



8VE/2U high density
DLC server

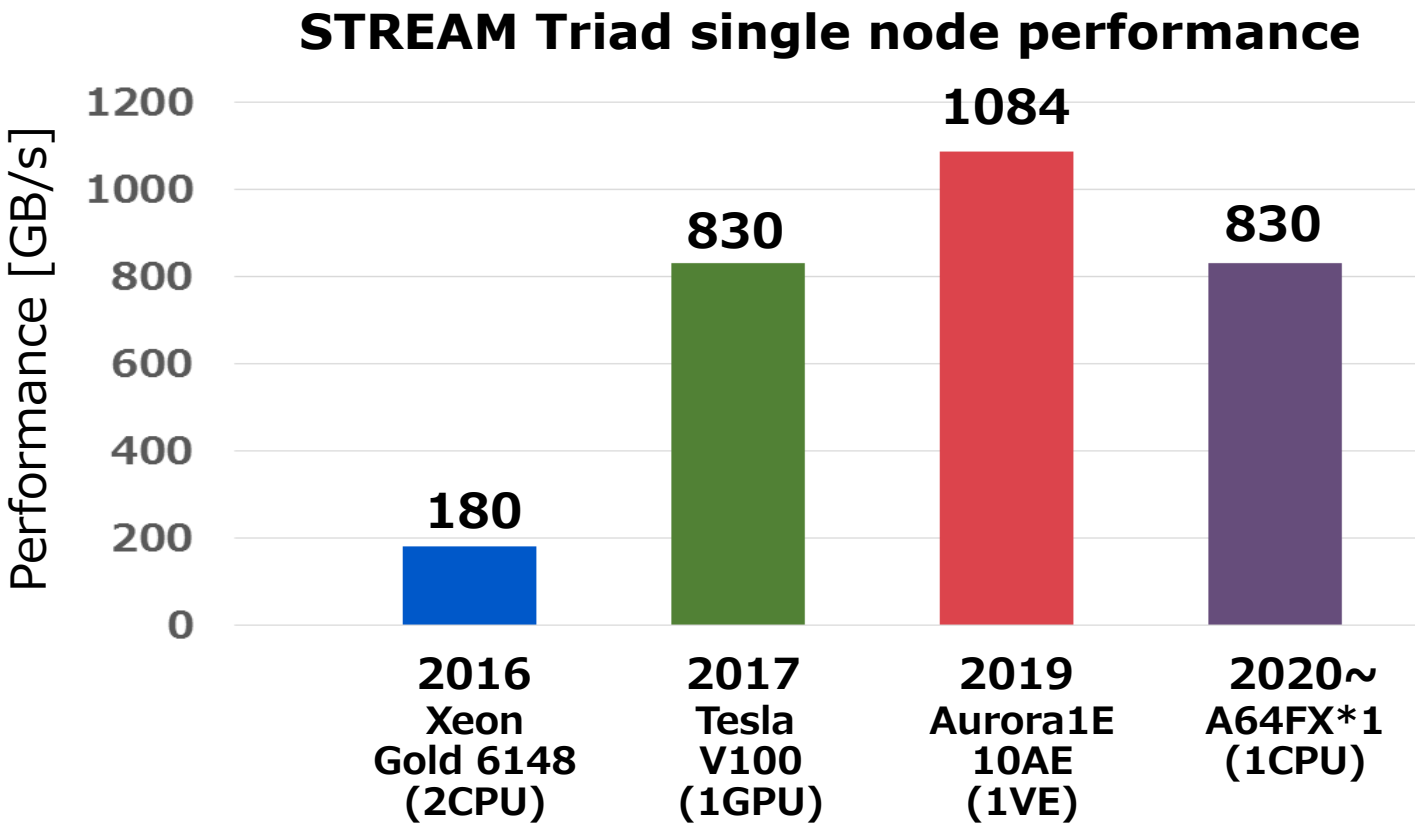
x18 servers
System integration



144VE/rack DLC
350TF/rack, 194TB/s/rack

Stream Benchmark

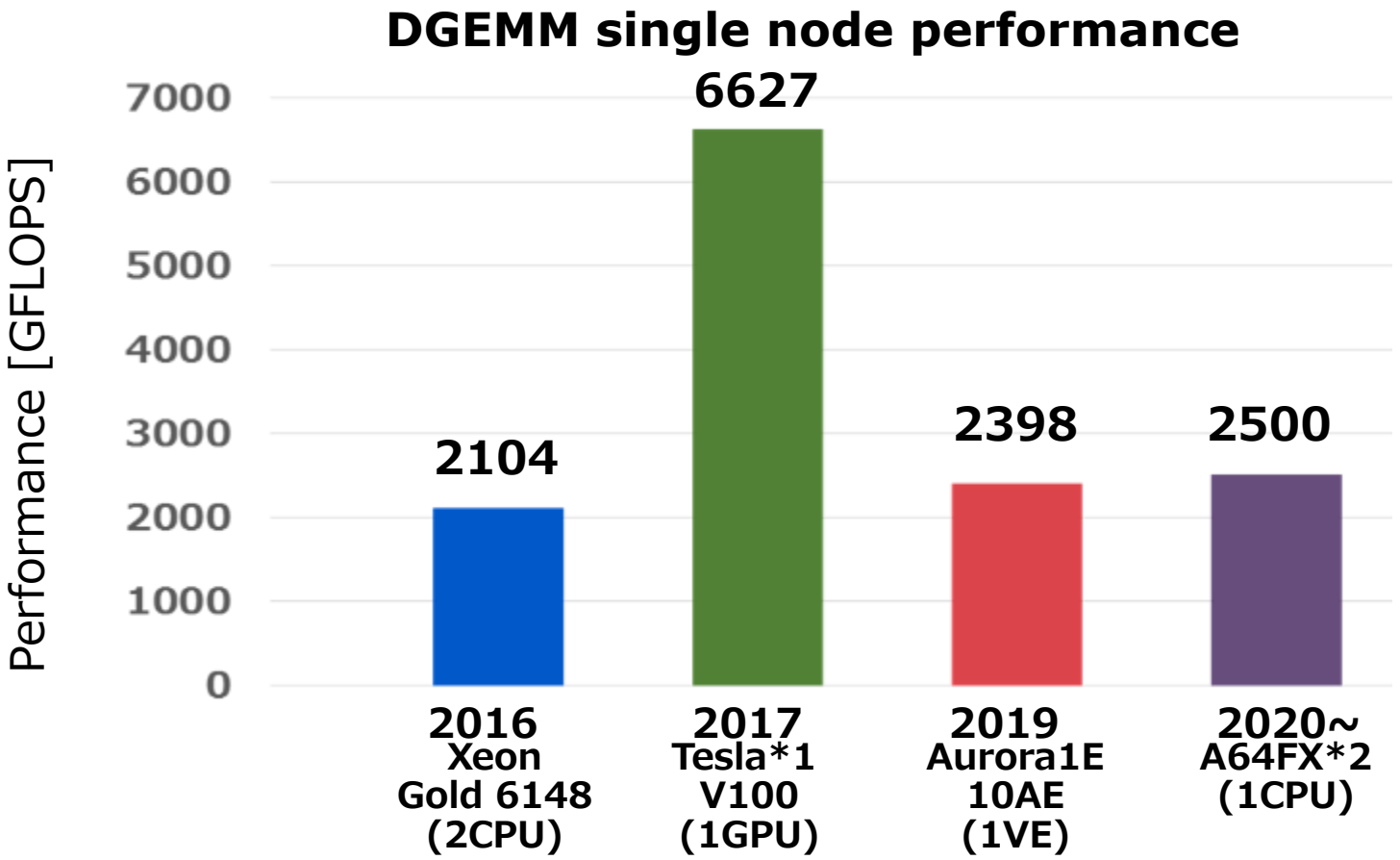
Aurora 1E performance is more than 30% higher than competitors



*1 The post-K project and Fujitsu ARM-SVE enabled A64FX processor
https://indico.math.cnrs.fr/event/4705/attachments/2362/2942/CEA-RIKEN-school-19013.pdf

DGEMM performance

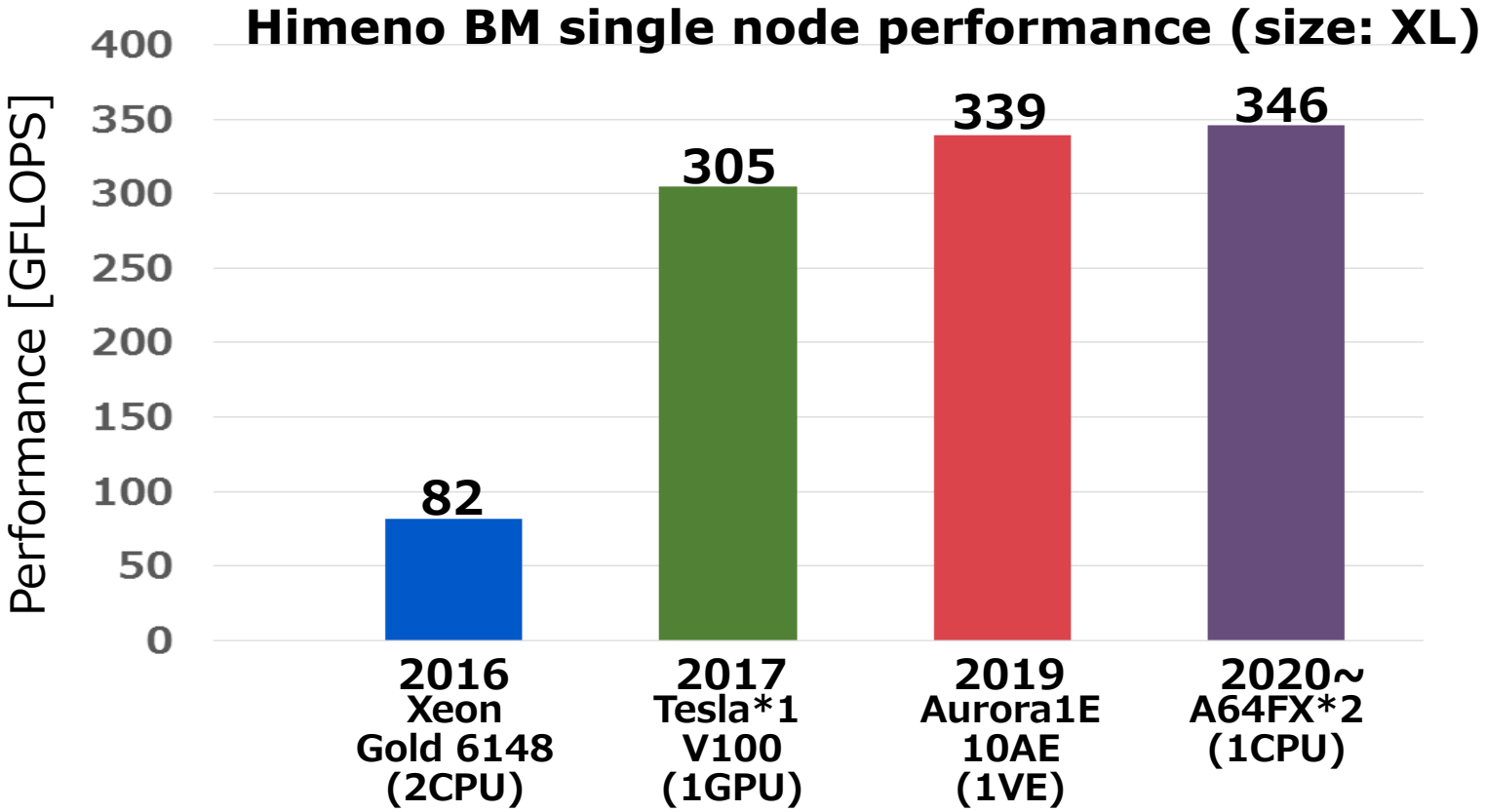
Aurora 1E performance is similar to Fugaku processor(A64FX)



*1 AMD NEXT HORIZON
http://ir.amd.com/static-files/ef99f84b-e1ad-4e12-8058-f3488f4c47b7
*2 The post-K project and Fujitsu ARM-SVE enabled A64FX processor
https://indico.math.cnrs.fr/event/4705/attachments/2362/2942/CEA-RIKEN-school-19013.pdf

Himeno Benchmark

Aurora 1E performance is similar to Fugaku Processor(A64FX)

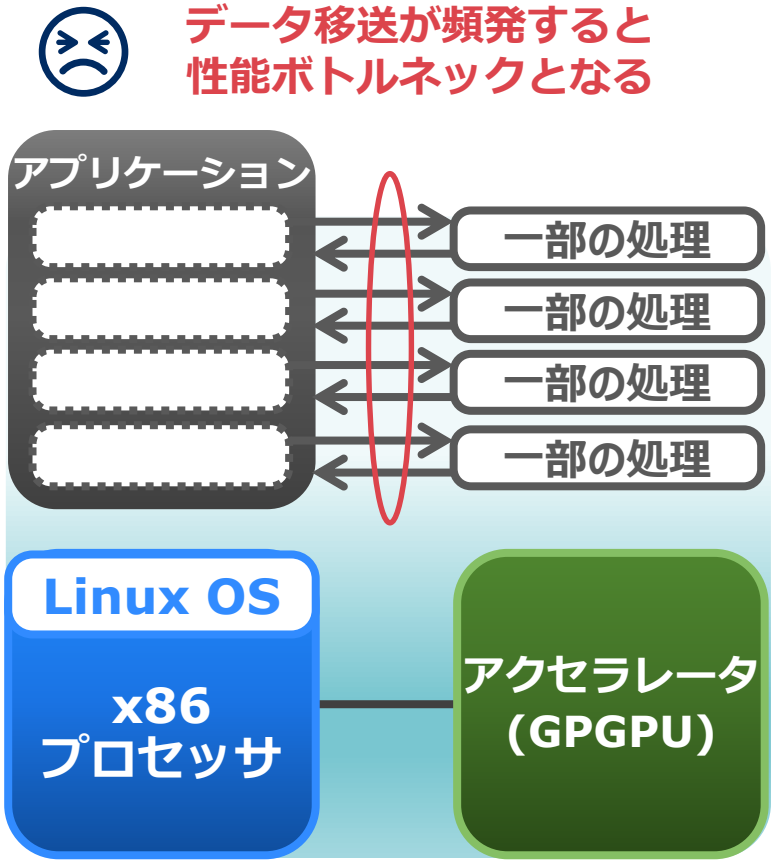


*1 Performance evaluation of a vector supercomputer SX-aurora TSUBASA
https://dl.acm.org/citation.cfm?id=3291728
*2 Supercomputer ”Fugaku” Formerly known as Post-K
https://www.fujitsu.com/global/Images/supercomputer-fugaku.pdf

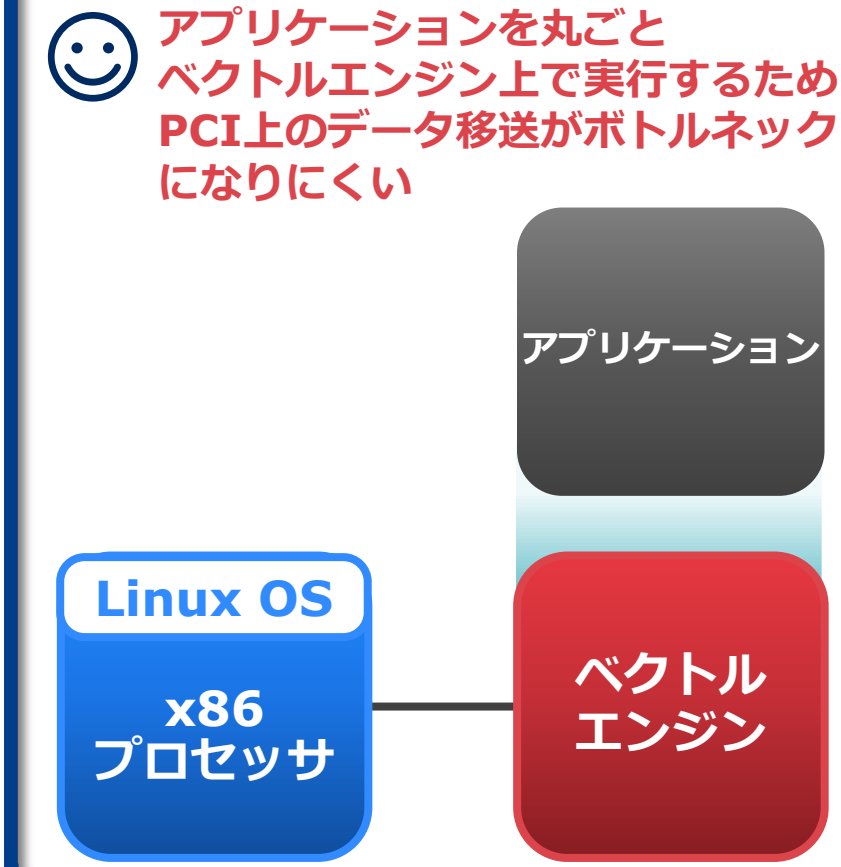
新しいプログラム実行モデル（基本）

GPGPUの持つ性能ボトルネックの解消

アクセラレータ型（GPGPU）

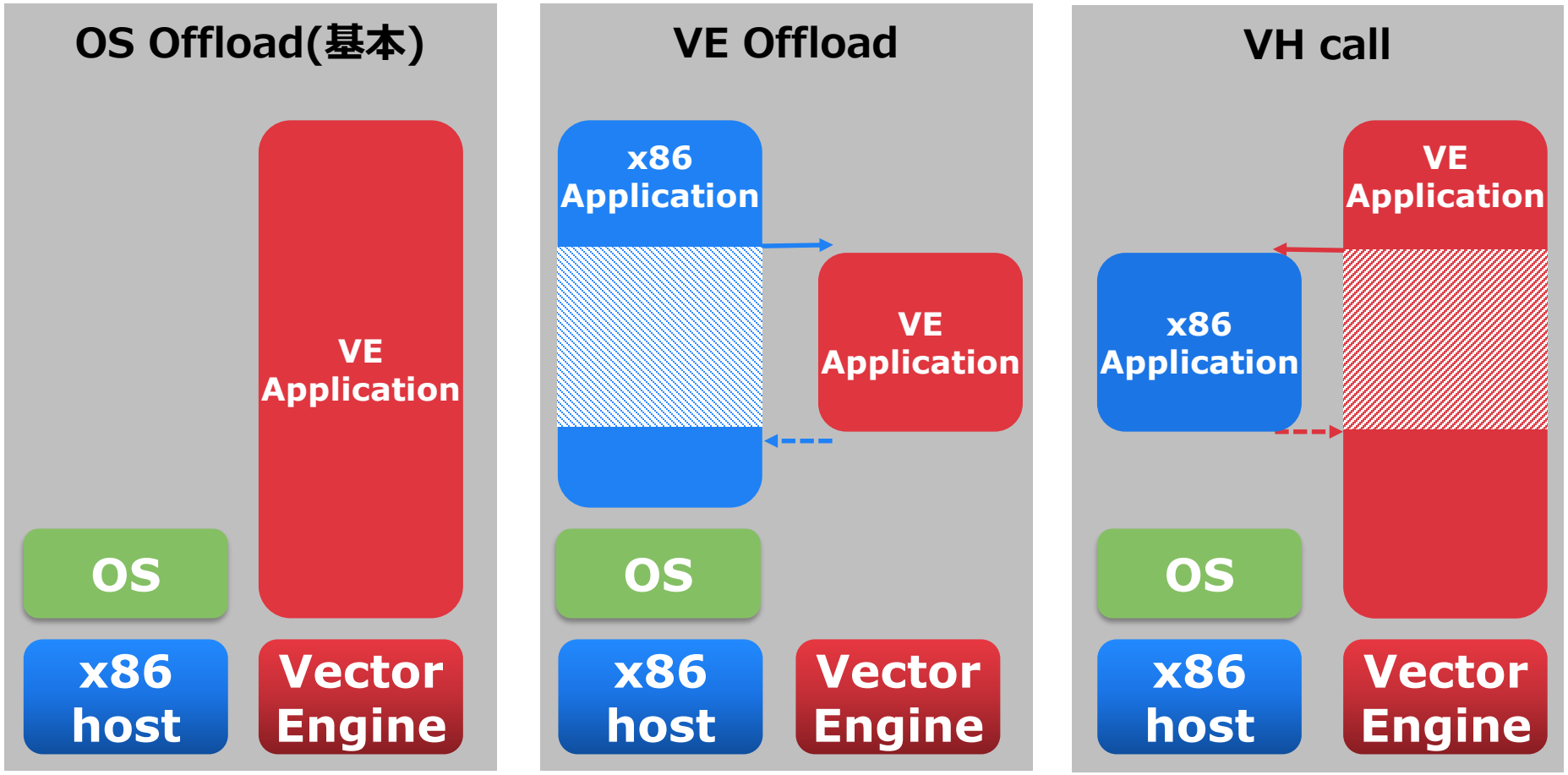


SX-Aurora TSUBASA



３つのオフロードモデル

アプリケーション特性に合わせて実行モデルを選択可能

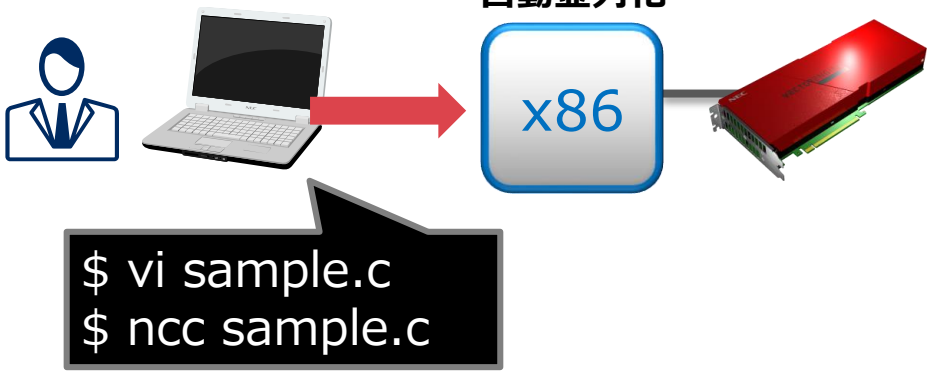


利用環境

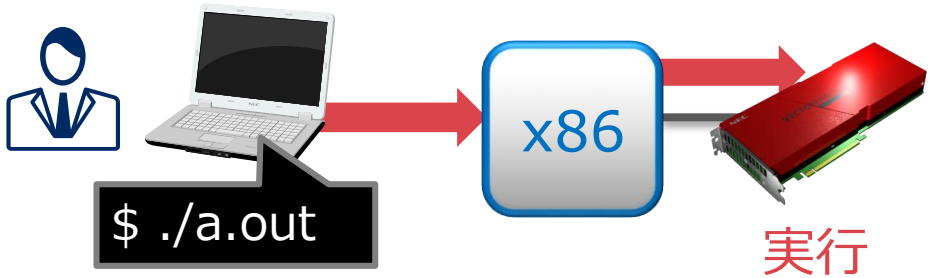
プログラミング環境

Vector Cross Compiler (Fortran/C/C++)

自動ベクトル化
自動並列化

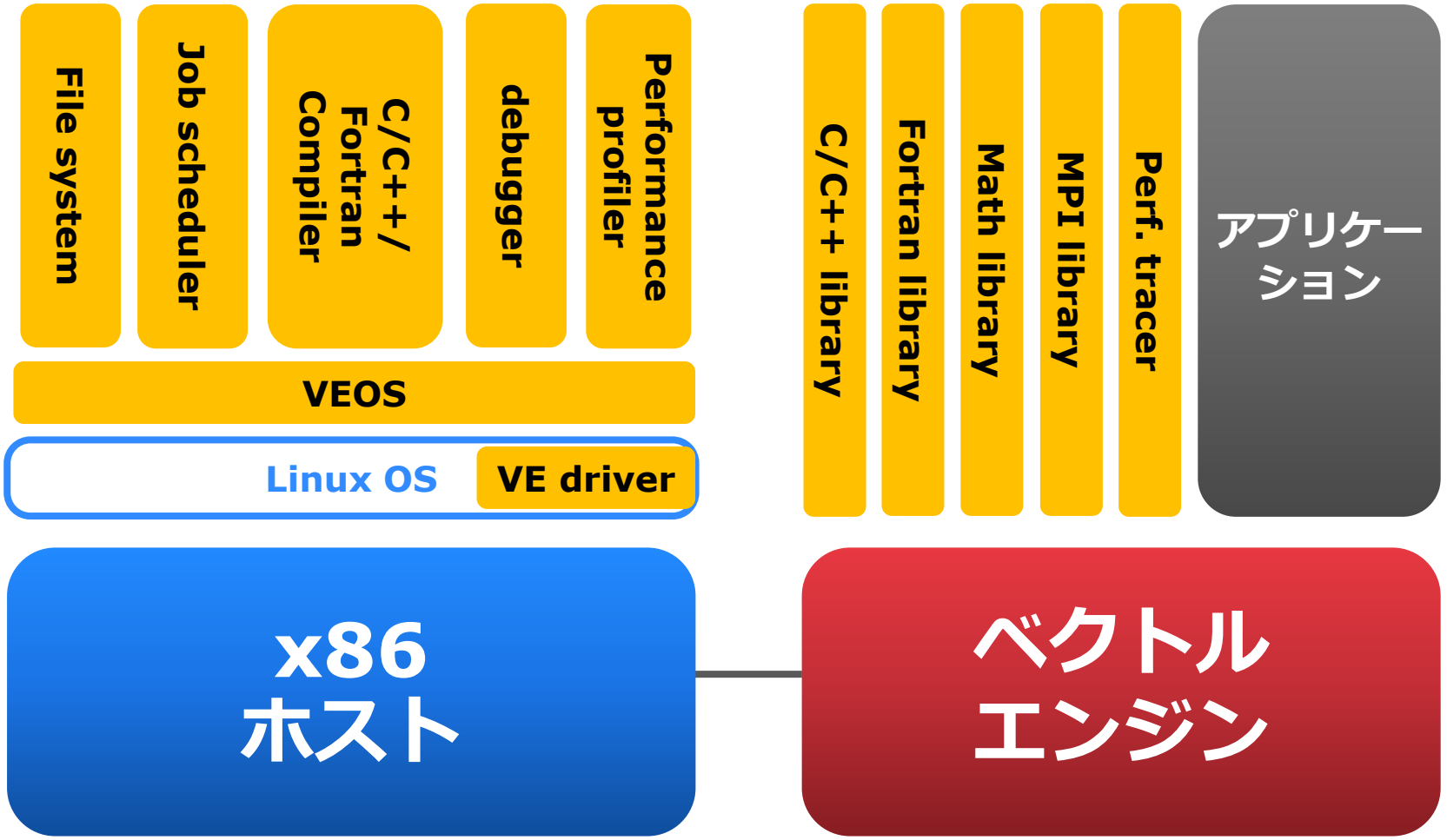


実行環境



Host OS:	RedHat Linux, Cent OS
Fortran:	F2003, F2008(partially)
C:	C11
C++:	C++14
OpenMP:	OpenMP4.5
MPI:	MPI3.1
Library:	glibc
	Numeric libraries (BLAS, FFT, Lapack, etc)
Tools:	gdb(GNU debugger)
	PTP(Eclipse Parallel Tools Platform)
	Ftrace(+viewer)/PROGINF

ソフトウェア・スタック

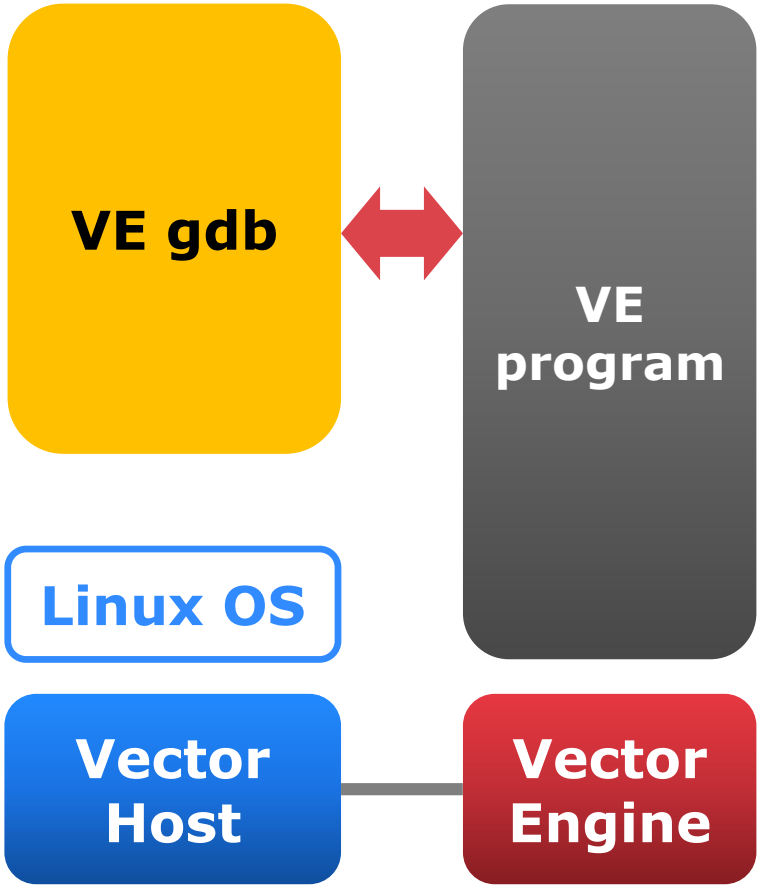


C/C++/Fortran プログラムのデバッグ環境

x86 gdbと同じ使用感で VE gdb は利用いただけます

- 基本的なデバッグ操作が可能
  - break pointsの設定
  - デバッガ上でプログラム実行
  - 実行中のプロセスのAttach
  - 変数値の表示、設定

- VE gdb 自体はVH（Vector Host)/Linux OS上で動作しています。



VEコンパイラ

これまでのSXシリーズで培った自動ベクトル化、自動並列化のコンパイラ技術を継承し、最新のハードウェア(VE)の性能を引き出します

```
FORMAT LIST

13: P------>  for (j = 0; j < m; j++) {
14: |V----->    for (i = 0; i < n; i++) {
15: ||            if (f[i] > 0.0) {
16: ||              a[j][i] = b[j][i] + sin(c[j][i]);
17: ||            }
18: ||      G     sum += d[i] + e[idx[i]];
19: |V-----     }
20: P------   }

DIAGNOSTIC LIST

 LINE              DIAGNOSTIC MESSAGE

  13: par(1801): Parallel routine generated.: fun$1
  13: par(1803): Parallelized by "for".
  14: vec( 101): Vectorized loop.
  18: vec( 126): Idiom detected.: Sum
```

外側ループを自動並列化し、内側ループを自動ベクトル化.

- 可変長ベクトル長（1~256）対応
- マスクベクトルによるif分のベクトル化
- 多くの組み込み関数、数値計算ライブラリ
- 連続、ストライド、間接アドレスのベクトル・メモリアクセス
- SUM,MAX,MINなどのReduction operation

性能情報　PROGINF and FTRACE

実行時オプション(PROGINF)やコンパイル・オプション(FTRACE)を指定することで、プログラムの性能情報を提供

PROGINF

```
    ********  Program  Information  ********
Real Time (sec)                :      121.233126
User Time (sec)                :      121.228955
Vector Time (sec)              :      106.934651
Inst. Count                    :   119280358861
V. Inst. Count                 :    29274454500
V. Element Count               :  6389370973939
V. Load Element Count          :  3141249840232
FLOP Count                     :  3182379290112
MOPS                           :     58637.969529
MOPS (Real)                    :     58635.323545
MFLOPS                         :     26251.407833
MFLOPS (Real)                  :     26250.223262
A. V. Length                   :       218.257559
V. Op. Ratio (%)               :        98.733828
L1 Cache Miss (sec)            :         0.639122
VLD LLC Hit Element Ratio (%)  :        73.051749
Memory Size Used (MB)          :     27532.000000

Start Time (date) : Sun Jul 22 20:09:51 2018 JST
End   Time (date) : Sun Jul 22 20:11:52 2018 JST
```

ベクトル実行時間、ベクトル演算率、平均ベクトル長などチューニングの目安となる多様な情報を採取・表示

FTRACE

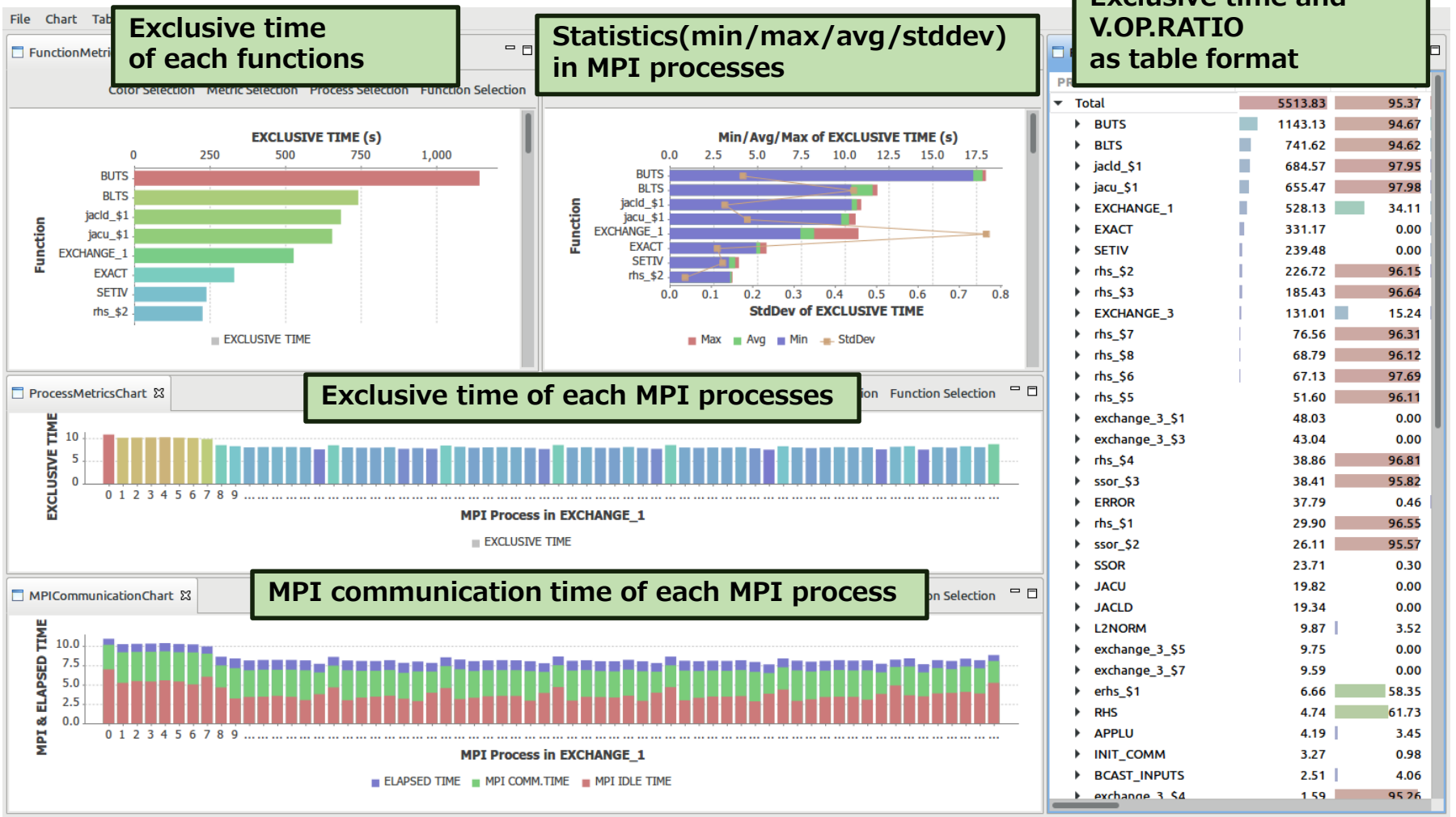
```
*----------------------*
  FTRACE ANALYSIS LIST
*----------------------*

Execution Date : Sun Jul 22 21:10:50 2018 JST
Total CPU Time : 0:02'02"434 (122.434 sec.)
```

FREQUENCY	EXCLUSIVE TIME[sec](%)	AVER.TIME [msec]	MOPS	MFLOPS	V.OP RATIO	AVER. V.LEN	VECTOR TIME	L1CACHE MISS	CPU PORT CONF	VLD LLC HIT E.%	PROC.NAME
512	58.110(47.5)	113.496	68380.5	30870.9	99.27	223.8	58.110	0.000	0.000	74.26	SUB1
510	31.763(25.9)	62.280	69474.6	31552.6	99.31	223.6	31.762	0.000	0.000	76.59	SUB2
2	9.056(7.4)	4527.963	3198.9	0.0	14.91	205.2	0.062	1.272	0.000	73.33	SUB3
459	7.732(6.3)	16.846	55793.0	23438.2	98.44	163.7	7.732	0.000	0.000	51.84	SUB4
459	7.037(5.7)	15.332	56239.0	26694.7	98.84	212.3	7.037	0.000	0.000	62.00	SUB5
2097152	6.667(5.4)	0.003	2816.0	322.1	22.88	256.0	0.282	0.283	0.000	100.00	SUB6
4	1.355(1.1)	338.818	19218.8	11094.9	98.75	243.9	1.355	0.000	0.000	0.82	SUB7
463	0.448(0.4)	0.968	57096.7	0.0	95.73	176.4	0.448	0.000	0.000	0.00	SUB8
1483	0.141(0.1)	0.095	18966.1	0.0	97.87	205.4	0.141	0.000	0.048	2.59	SUB9
2099270	0.122(0.1)	0.000	3056.8	17.2	0.00	0.0	0.000	0.000	0.000	0.00	SUB10
51	0.001(0.0)	0.017	667.2	0.0	0.00	0.0	0.000	0.001	0.000	0.00	SUB11
1	0.000(0.0)	0.292	444.7	0.6	0.01	8.0	0.000	0.000	0.000	0.00	MAIN_
4201150	122.434(100.0)	0.029	58071.6	25992.7	98.59	218.3	106.929	1.558	0.048	73.05	total

Ftrace Viewer

- GUI プロファイラー (Ftrace GUI-frontend)
- OpenMP, MPIの両方を使ったプログラムに対応



Numeric Library Collection

科学技術計算で頻出する機能をベクトルエンジン向けに最適化した数学ライブラリのコレクション

特長

- ベクトルエンジン向けに高度に最適化
- 幅広い機能を網羅
  - 線形代数、フーリエ変換、疑似乱数生成、統計機能、…
- デファクトスタンダードのライブラリを提供
  - BLAS、LAPACK、ScaLAPACKなど

NLCに含まれるライブラリ

- ASL
- BLAS / CBLAS
- LAPACK
- ScaLAPACK
- HeteroSolver
- Stencil Code Accelerator
- SBLAS

新機能

他社ライブラリ製品との比較

- NLCは他社ライブラリ製品より豊富な機能を提供
- 今後はディープラーニング向けの機能も拡充予定

		NLC	MKL	CUDA
線形代数	密行列計算 基本演算、連立一次方程式、固有値方程式など	✓	✓	✓
	疎行列計算 基本演算、連立一次方程式、固有値方程式など	✓	✓	✓
変換	フーリエ変換	✓	✓	✓
	Real-to-Real (DCT, etc.)	✓	✓	
	ラプラス、ウェーブレット変換	✓		
統計	疑似乱数生成	✓	✓ w/o MPI	✓ w/o MPI
	多変量解析、回帰分析など	✓		
その他	ソート	✓		
	特殊関数	✓		
	微分・積分など	✓		
	ステンシル計算	✓		
	ディープラーニング		✓	✓

HeteroSolver -1/2-

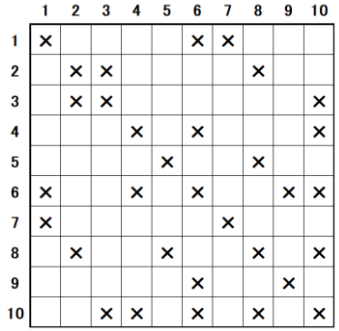
疎行列連立一次方程式直接法ソルバ
ベクトルエンジン(VE)とベクトルホスト(VH)が連携して求解

疎行列直接法解法の処理

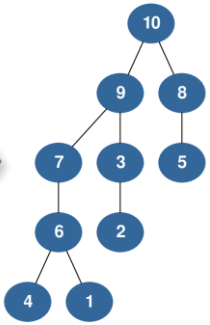
$$Ax = b$$

- 前処理
  - 疎行列Aを分析

  ← ベクトル性能を出しにくい
- 数値分解処理
  - LU分解、LDL^t分解
- 求解処理
  - 解xを算出



行列 A



消去木

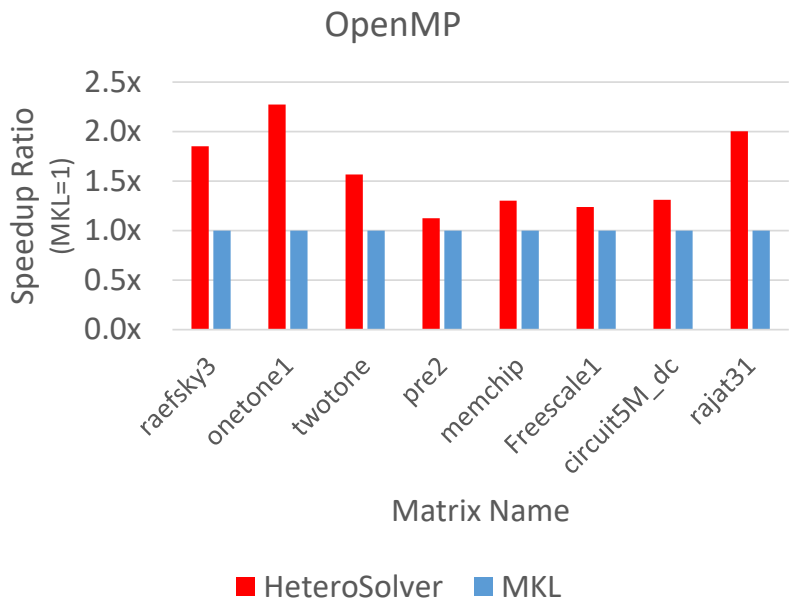
VE-VH連携を使った最適化

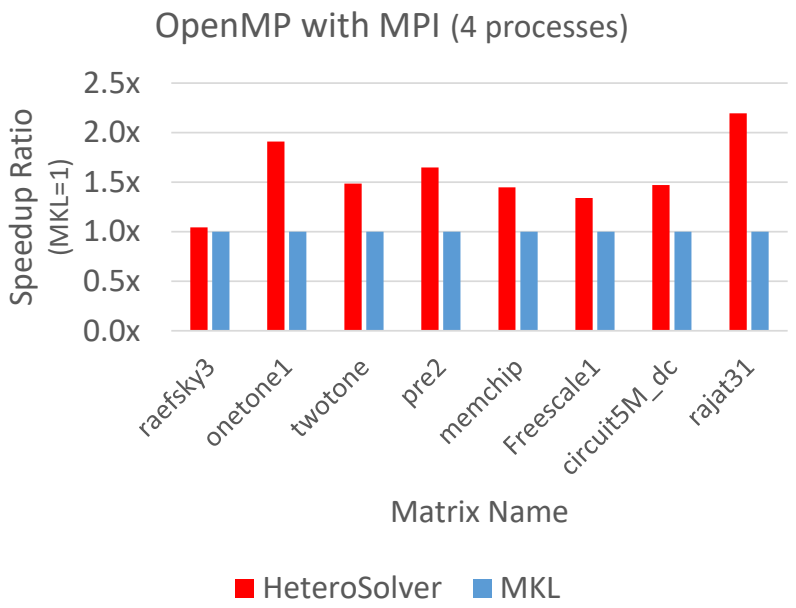
- ライブラリ内部で前処理に必要なデータをベクトルホスト(VH)に転送して実行

HeteroSolver -2/2-

HeteroSolverの利用によりSkylake(40コア)より高速に求解が可能に

ベンチマーク





	実行マシン
HeteroSolver	SX-Aurora TSUBASA (1.4 GHz, 8 Cores)
MKL	Xeon Gold 6148 x 2 sockets (Skylake 2.4 GHz, 40 cores)

使用した行列: SuiteSparse Matrix Collection (https://sparse.tamu.edu/)からダウンロード

Stencil Code Accelerator

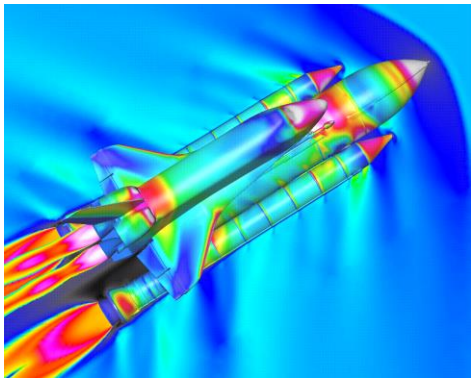
ステンシル計算の応用分野

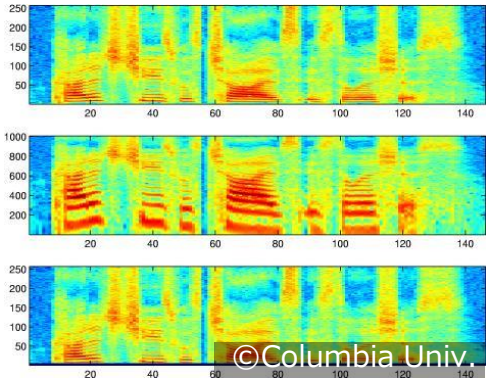
- 科学技術シミュレーション
  - 流体解析
  - 熱伝導解析
  - 電磁場解析

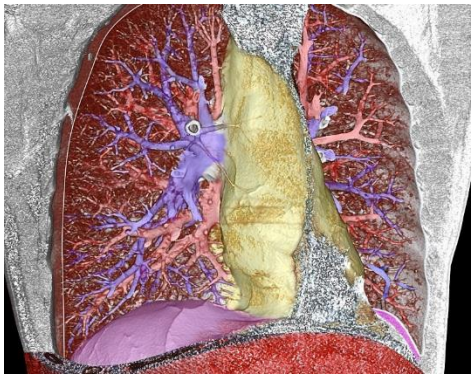
  など
- 信号処理
  - 音響、ソナー
  - レーダ、無線通信

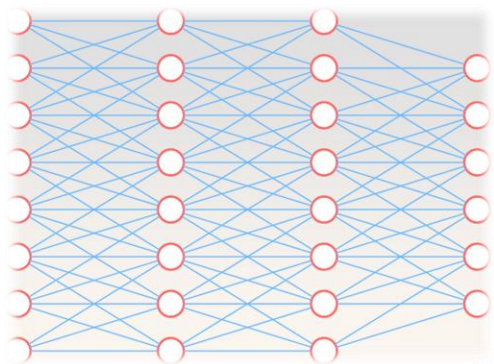
  など
- 画像 / 3Dデータ処理
  - 画像編集・加工
  - データ圧縮
  - 認識
  - 医療診断 (生検, CT, MRI, …)

  など
- 機械学習
  - ディープラーニング (CNN)









Stencil Code Accelerator: コード例 -1/2-

ステンシル計算の例

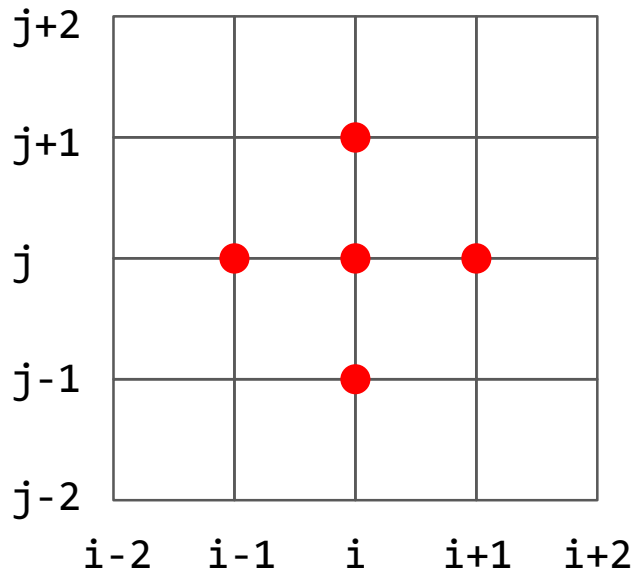
```
real :: a(0:ni+1,0:nj+1)
real :: b(1:ni,1:nj)
    :
do itr=1,maxitr
    :
   do j=1,nj
   do i=1,ni
      b(i,j)=0.25*(  &
          a(i  ,j-1) &
         +a(i-1,j  ) &
         +a(i+1,j  ) &
         +a(i  ,j+1))
   end do
   end do
    :
   a(:,:)=b(:,:)
end do
```

ラプラス方程式

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

離散化
有限差分

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



Stencil Code Accelerator: コード例 -2/2-

ステンシル計算の例 – SCA利用

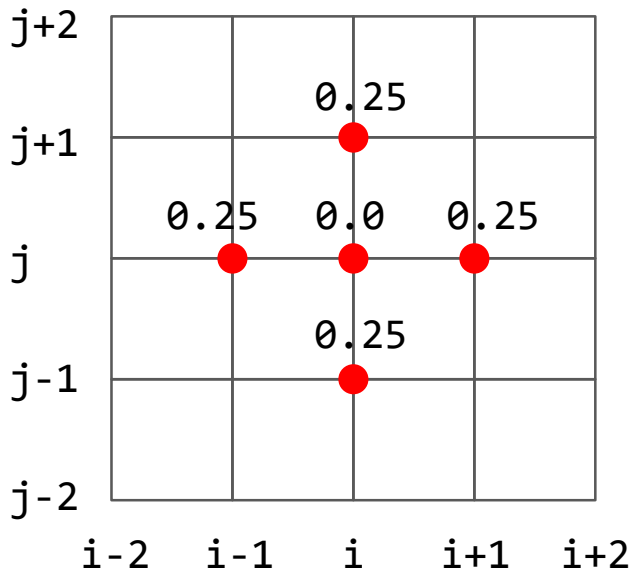
```
real :: a(0:ni+1,0:nj+1)
real :: b(1:ni,1:nj)
real,parameter :: c(5)=(/ &
   0.25,0.25,0.0,0.25,0.25/)
    :
do itr=1,maxitr
    :
   call sca_compute_with_1x1ya_s( &
      ni,nj,1, &
      ni+2,nj+2,a(1,1), &
      ni,nj,b(1,1),c,0.0)
    :
   a(:,:)=b(:,:)
end do
```

ラプラス方程式

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

離散化
有限差分

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



Stencil Code Accelerator: ステンシルの形状

56ケースのステンシル形状に対応した機能を提供

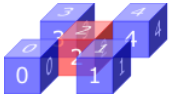
ステンシルの形状

- 対応しているステンシルの形状
  - {X,Y,Z}-方向

  - {XY,XZ,YZ}-平面

  - {XY,XZ,YZ}-軸方向

  - {XY,XZ,YZ}-対角方向



  - XYZ-立体

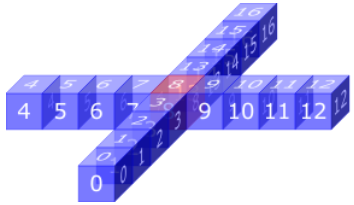
  - XYZ-軸方向

- 各形状について以下のサイズに対応
  - 1

  - 2

  - 3

  - 4

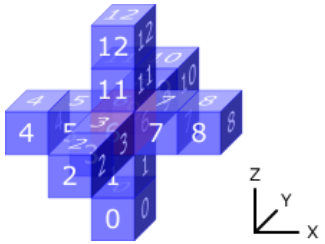


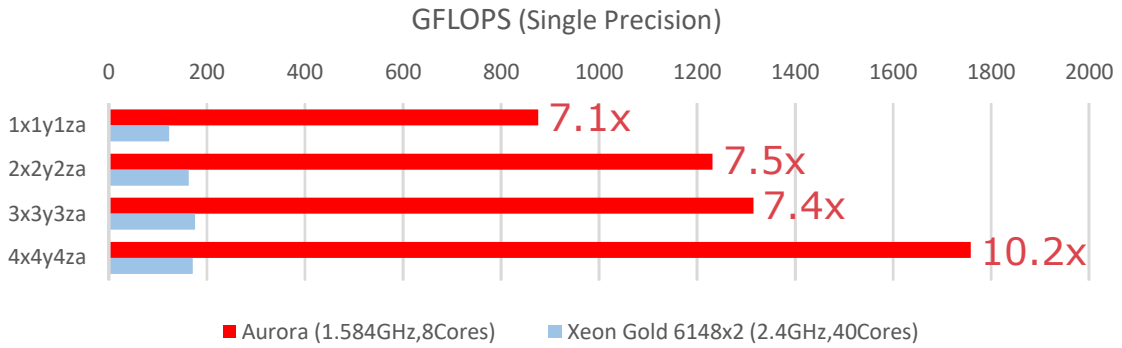
Stencil Code Acceleratorの性能

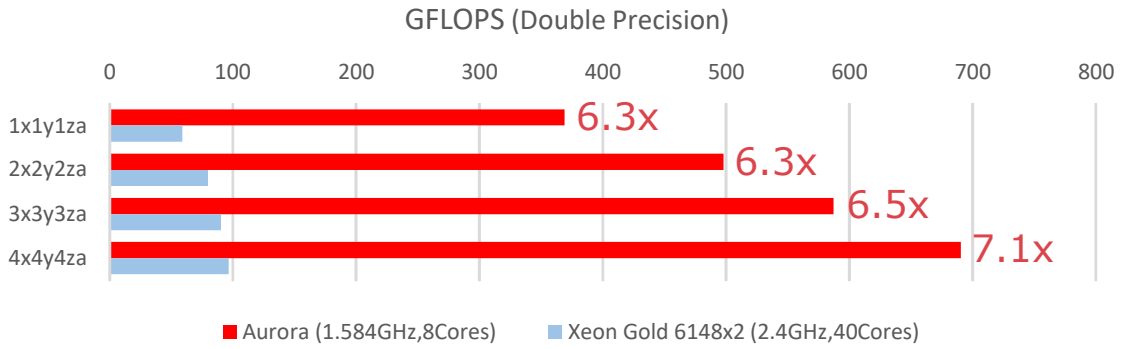
SCAの利用によりXeon(40コア)より6.3～10.2倍も高速

ベンチマーク結果

- ステンシルの形状: XYZ-軸方向, サイズ 1～4　← 科学技術計算での頻出ケース
- データサイズ: 1024 x 1024 x 1024
- プロセッサ:
  - Aurora 1.584 GHz, 8コア
  - Xeon Gold 6148 x 2 ソケット (Skylake 2.4 GHz, 40コア)





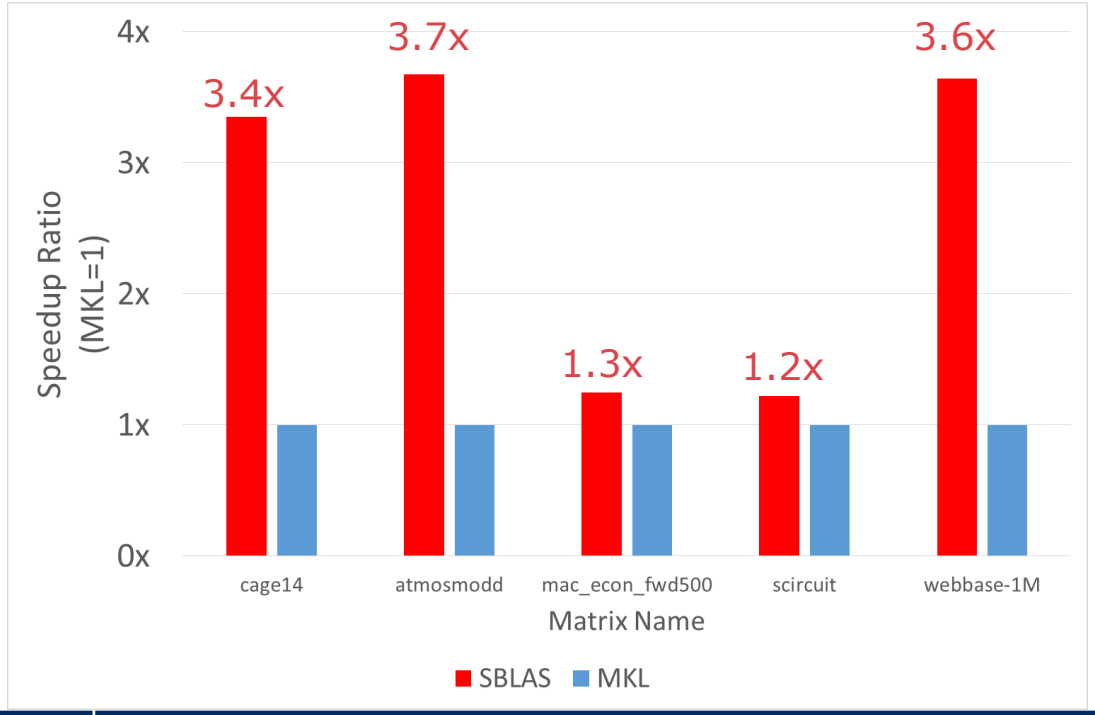


◆ 上記性能は、ベクトルレジスタをキャッシュとして利用しメモリ負荷を削減することにより達成

SBLAS

疎行列ベクトル積の機能を新規に追加、MKLより1.2倍～3.7倍高速

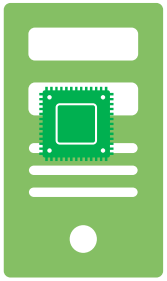
ベンチマーク結果



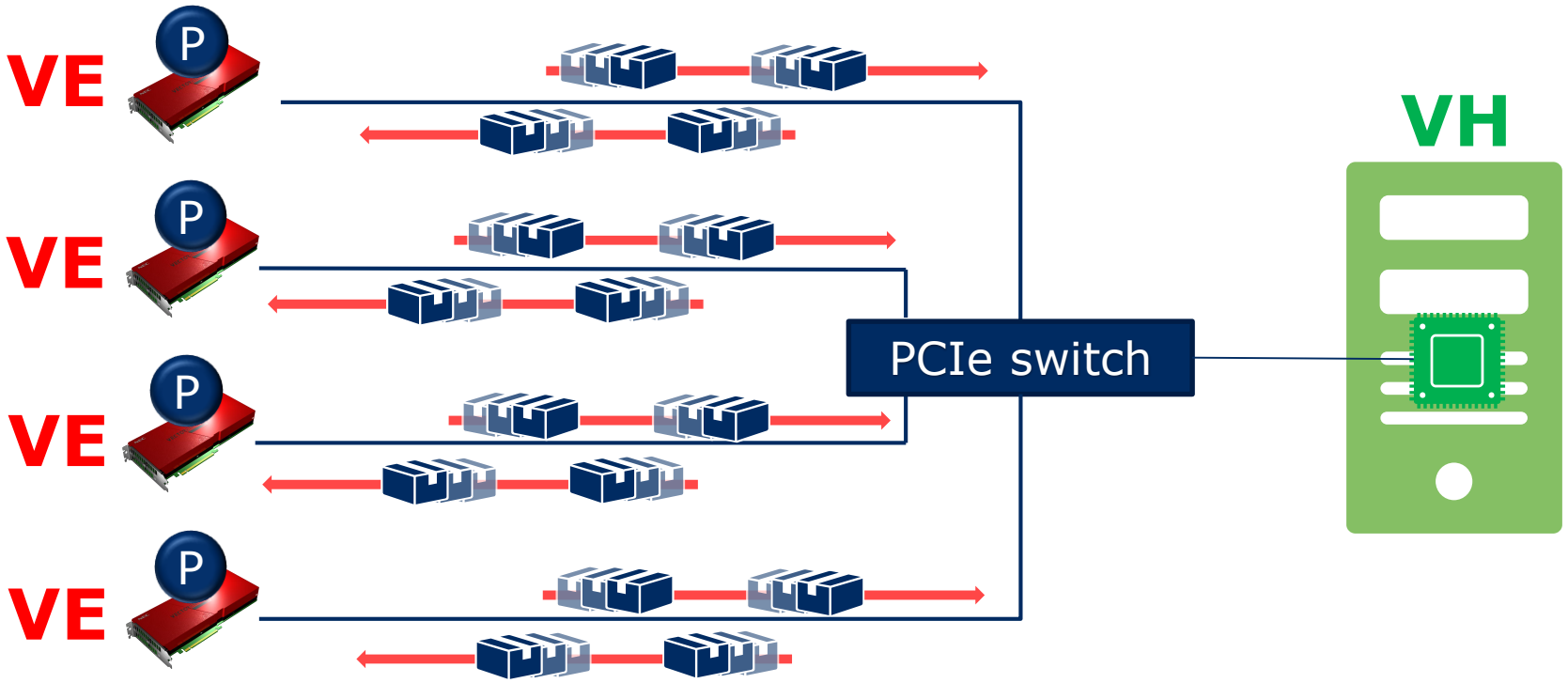
	実行マシン
SBLAS	SX-Aurora TSUBASA (1.4 GHz, 8 Cores)
MKL	Xeon Gold 6132 x 2 sockets (Skylake 2.6 GHz, 56 cores)

使用した行列: SuiteSparse Matrix Collection (https://sparse.tamu.edu/)

Communication between VE and VH

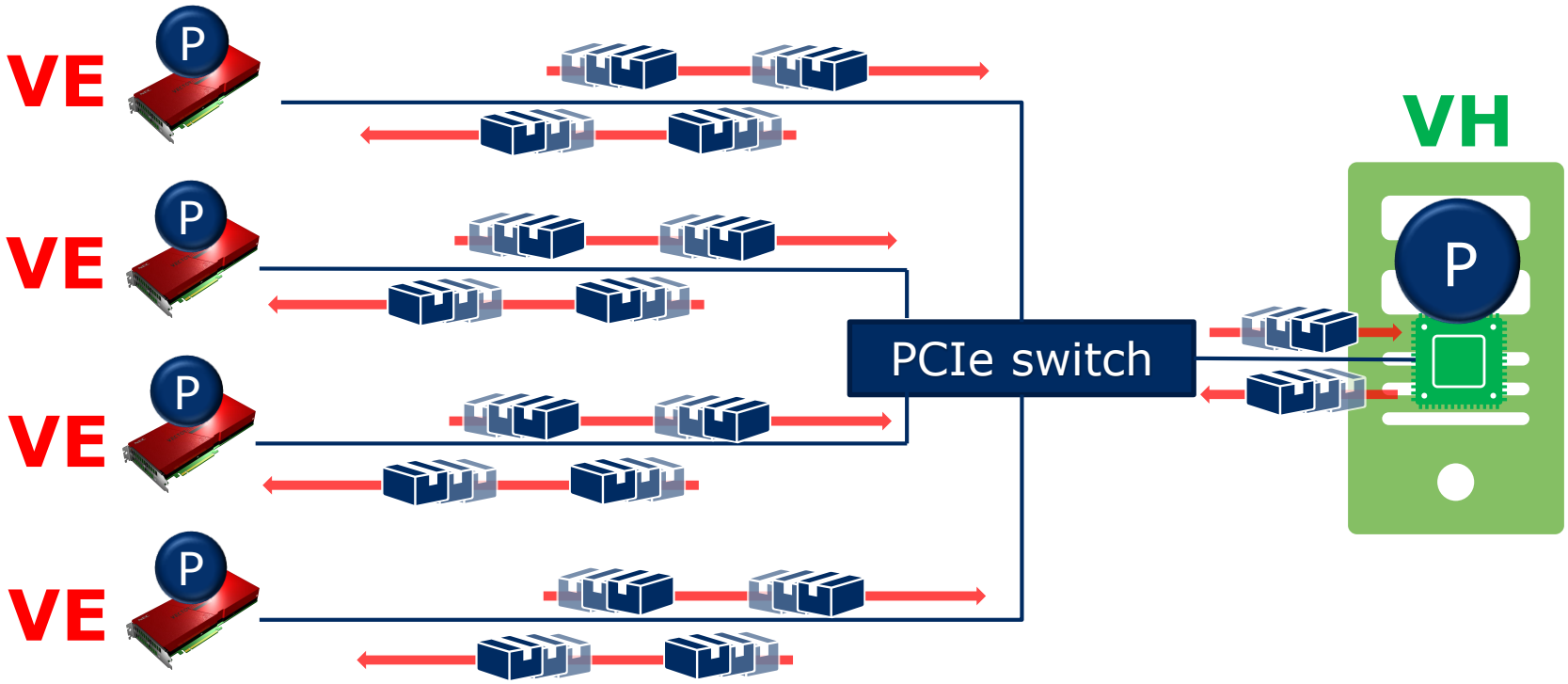


Auroraの一般的なVE間MPI通信　（OS offloadモデル）



P Process

Hybrid MPI on Aurora



P Process

Result of HPL

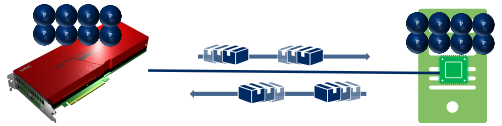
*8 procs on **VE***

1867 Gflops

*8 procs on **VH***

1430 Gflops

Hybrid MPI



*16 procs on **VE** **and** **VH***

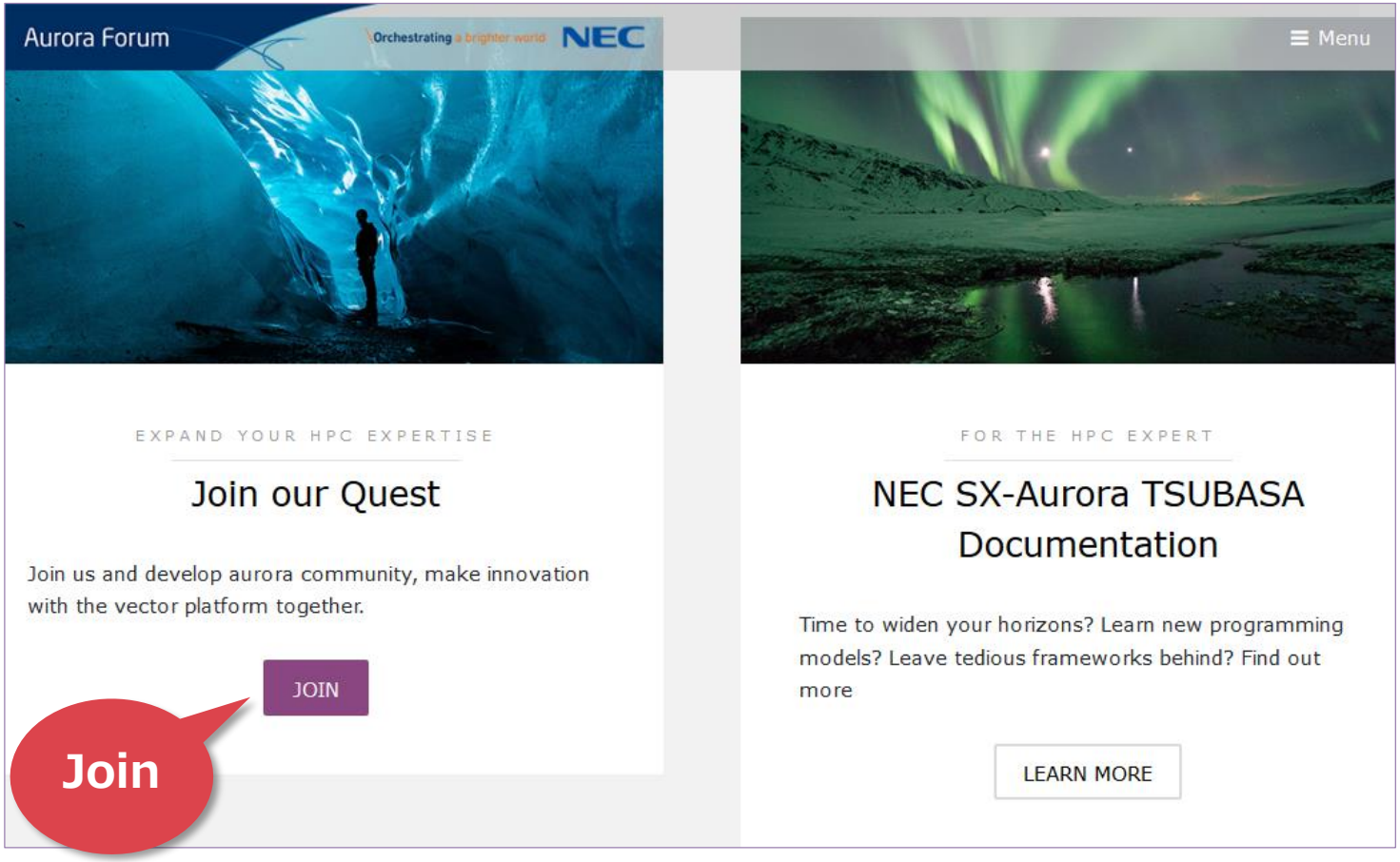
2830 Gflops

Aurora Forum Website

製品マニュアル・技術情報はAurora Forum Websiteにて公開中

- URL - https://www.hpc.nec/

Aurora Forum



ガイド類

プログラム実行クイックガイド

- https://www.hpc.nec/documents/guide/pdfs/ProgramExecutionQuickGuide_J.pdf

NLC (NEC Numeric Library Collection) ユーザーズガイド

- https://www.hpc.nec/documents/sdk/SDK_NLC/UsersGuide/main/ja/index.html

ベクトル化／並列化の Fortran/C/C++ コンパイラガイド

- https://www.hpc.nec/api/v1/forum/file/download?id=n7Yhz4 (Fortran)
- https://www.hpc.nec/api/v1/forum/file/download?id=nDkhWe (C/C++)

チューニング・ガイド

- https://www.hpc.nec/api/v1/forum/file/download?id=LbGhrV

VE offload ～VH(x86)からVEで動作する関数ライブラリを呼び出す方法

- https://veos-sxarr-nec.github.io/veoffload/index.html

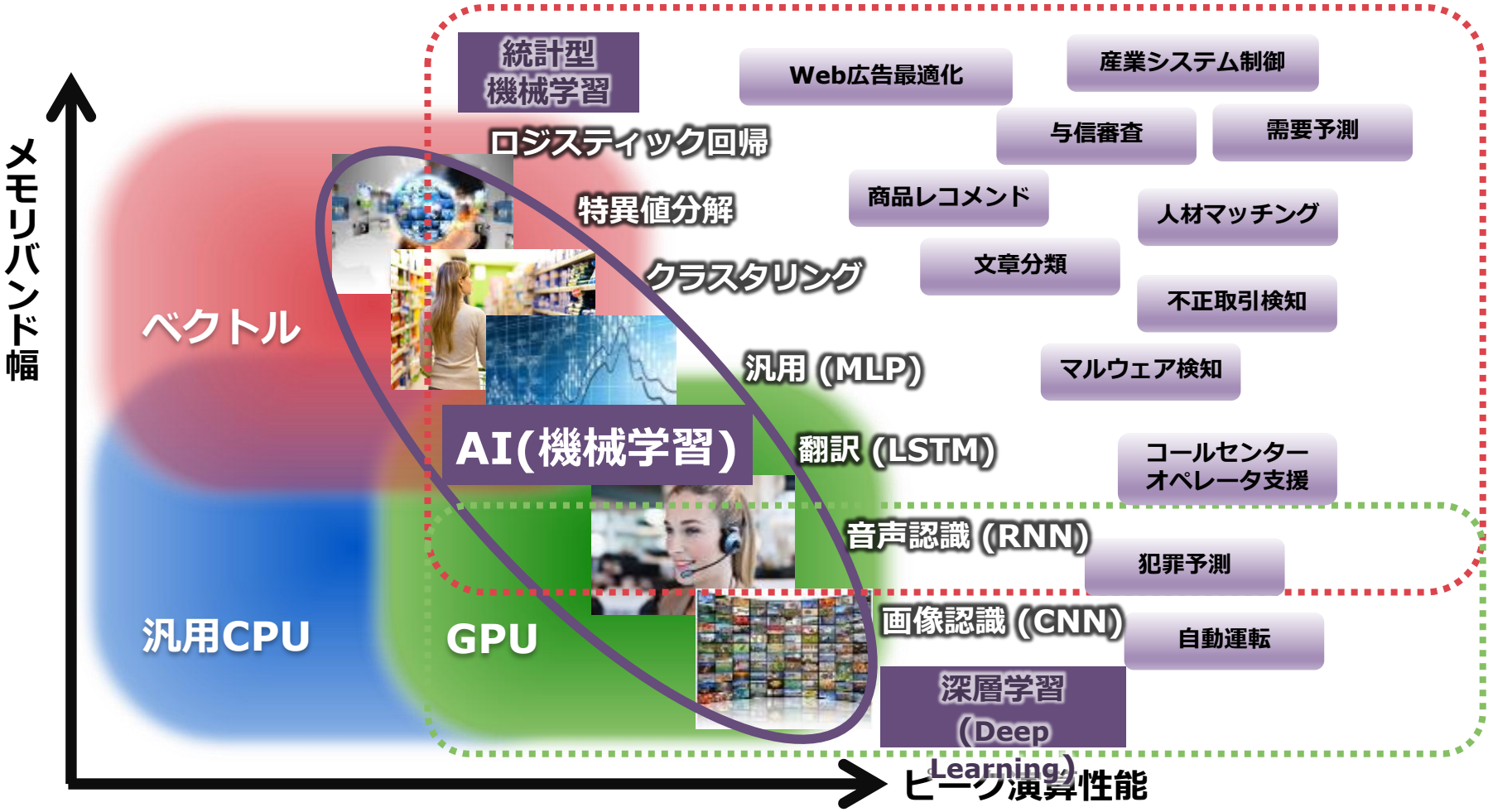
Hybrid MPI ~Hybrid MPIの実行方法

- https://www.hpc.nec/documents/mpi/pdfs/g2am01-NEC_MPI_User_Guide_ja.pdf
  （MPIユーザガイド中にHybrid MPIの実行方法等も記載）

AI/BDAでの活用

機械学習の特性、及び応用例

統計数理型機械学習は演算性能とメモリバンド幅のバランスが重要で、Auroraが得意とする分野



機械学習(Python/Spark)を高速化するミドルウェア：Frovedis*

- Python/Sparkからベクトルを意識せず利用可能なミドルウェア
- SX-Aurora TSUBASAの高いメモリ性能を生かし、機械学習を高速化

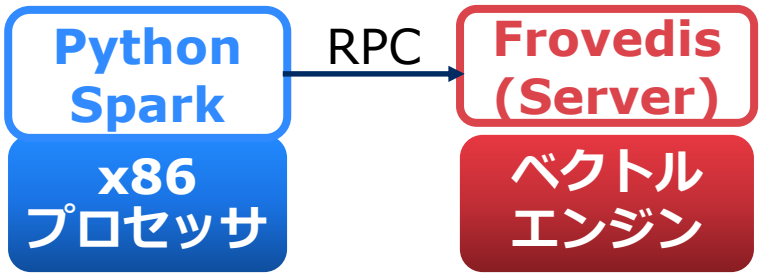
※ FRamework Of VEctorized and DIStributed data analytics

データ分析で利用されるPythonライブラリ（機械学習：scikit-learn, データフレーム：pandas）に実装されているアルゴリズムをサポート

- 機械学習ライブラリ (Python scikit-learn、Spark Mllib I/F互換)
- Data Frameライブラリ (Python pandas, Spark Dataframe)
- NECでOSS化しており、下記のリンク先で公開中
  https://github.com/frovedis

既存ライブラリ(scikit-learn, MLlib)と同じ形式で呼び出すことが可能

- importするモジュールを切り替えるだけ



Frovedisを使うPythonプログラムの概要

```
from frovedis.exrpc.server import FrovedisServer # frovedis

# インポートするライブラリを切り替える
from frovedis.mllib.linear_model import LogisticRegression # frovedis
#from sklearn.linear_model import LogisticRegression # sklearn
...

# Frovedisサーバーを起動する
FrovedisServer.initialize("mpirun -np 4
{}".format(os.environ['FROVEDIS_SERVER']))

# 以下はscikit-learnと同じ
clf = LogisticRegression(random_state=0, C=10.0, max_iter=10000).fit(X, y)
score = 1.0 * sum(y == y_pred) / len(y)

# Frovedisサーバーを終了する
FrovedisServer.shut_down()
```

Frovedis でサポートするアルゴリズム

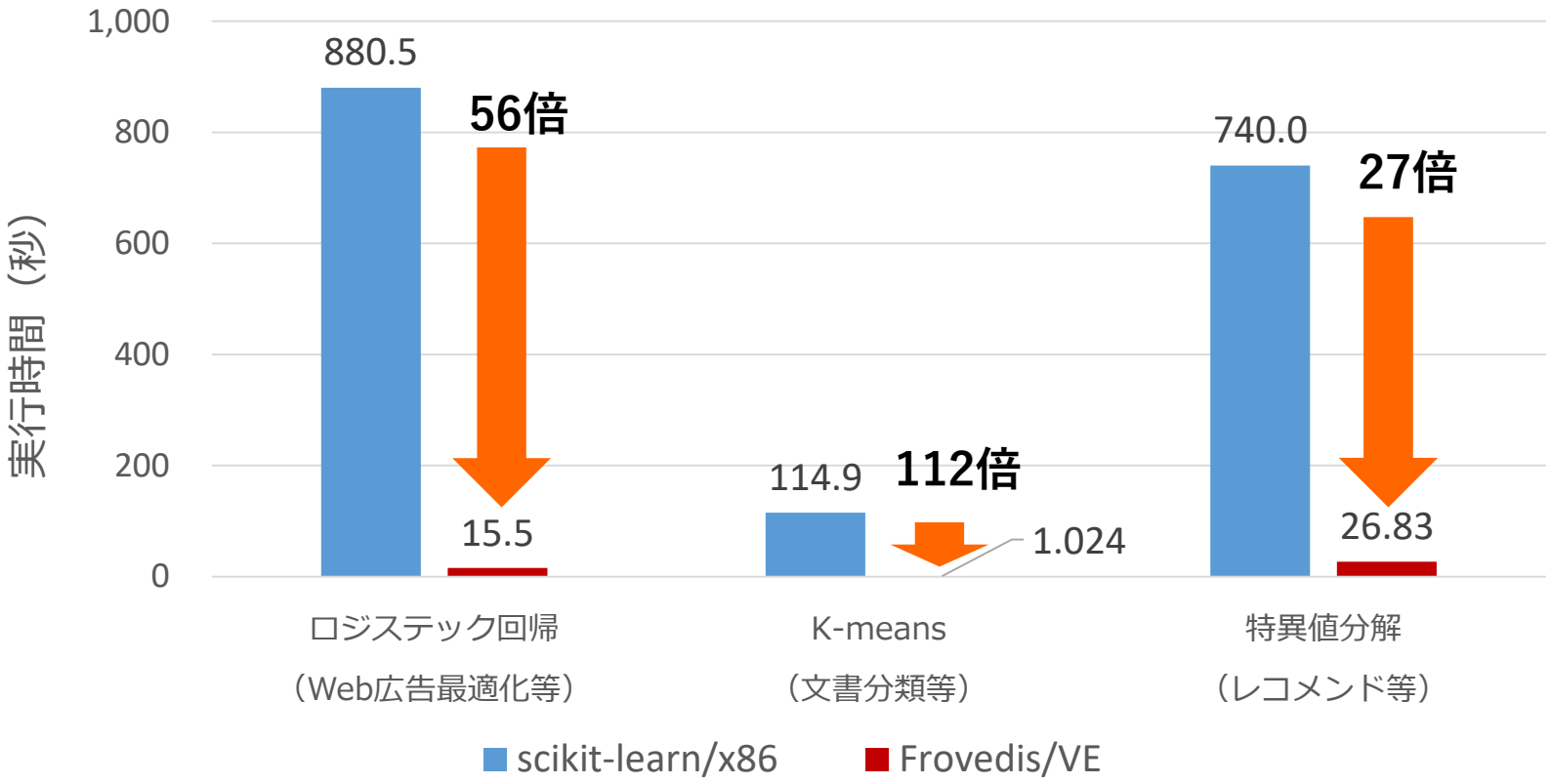
scikit-learn（機械学習）

- linear_model.LogisticRegression
- linear_model.LinearRegression
- linear_model.Lasso
- linear_model.Ridge
- svm.LinearSVC
- cluster.KMeans
- tree.DecisionTreeRegressor
- tree.DecisionTreeClassifier
- naive_bayes.MultinomialNB
- naive_bayes.BernoulliNB
- decomposition.TruncatedSVD

Frovedisの基本性能

x86上のscikit-learnとの実行時間を比較：

- Xeon (Gold 6126) 1ソケット 対 ベクトルエンジン1基
- Frovedisライブラリの効率の良さと、ベクトル型コンピュータを用いた高速化により、**25倍～110倍**の高速化を確認

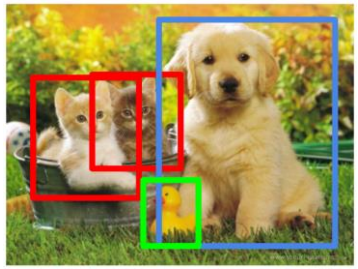


＊ ロジスティック回帰はCriteoが公開しているデータの一部(4.8GB)。K-meansはgisett_scaleデータの一部(240MB)。
特異値分解は英文Wikipediaの一部(5.2GB)。いずれもIO時間を含む。

深層学習の適応性

ニューラルネットワーク種類とデータ特性

CNN



data with local features
(ex. image)

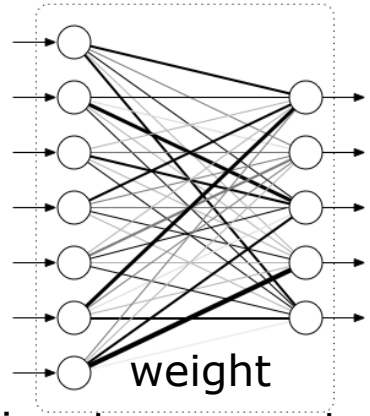
RNN(LSTM)

time series data
(ex. speech, sensor)

MLP

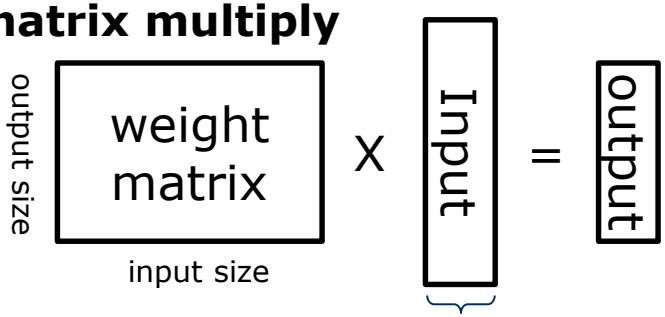
generic data
(ex. transaction log)

全結合層 (MLPのコア部分)



全結合層の計算:

matrix multiply



メモリバンド幅が性能を決定

(**load of weight is dominant** when minibatch is small)

高メモリバンド幅のSX-AuroraはMLPに最適!!

深層学習を利用したマルウェア検出

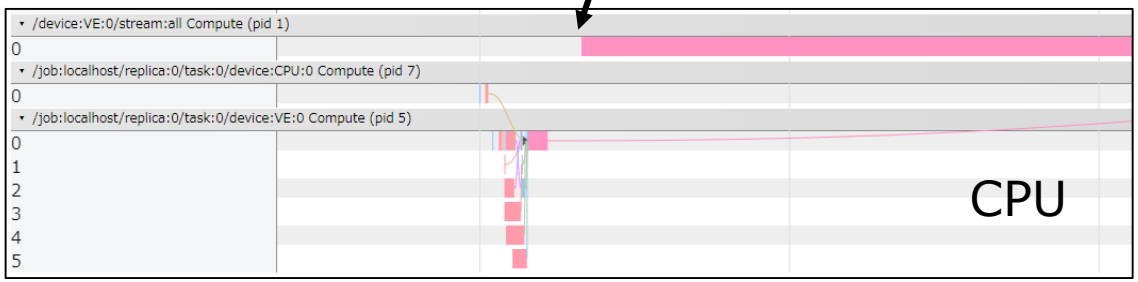
バイナリファイルからマルウェアを検出する小規模モデルのトレーニング

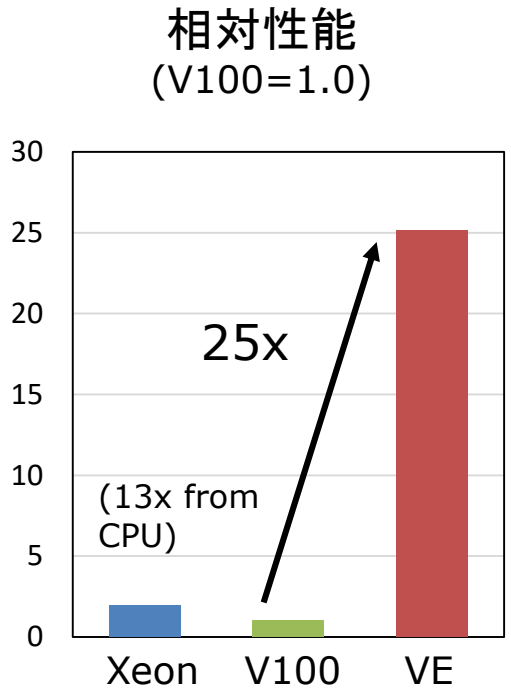
- NECマルウェア検出サービスで利用されているモデルをTensorFlowに移植して、CPU,GPU,ベクトルエンジンで評価

ベンチマーク環境

- プラットフォーム:
  - CPU: Xeon Gold 6126, 1.3TF, 120GB/s
  - GPU: V100(PCIe), 14TF, 900GB/s
  - VE: Type 10B, 4.3TF, 1.2GB/s
- データサイズ: 760MB

SX-Auroraのタイムライン






Orchestrating a brighter world
NEC