MN-Core アーキテクチャと開発環境

2024/12/19 FOCUS講習会

Preferred Networks, Inc.

Preferred Networks（PFN）会社概要

ミッション： 現実世界を計算可能にする

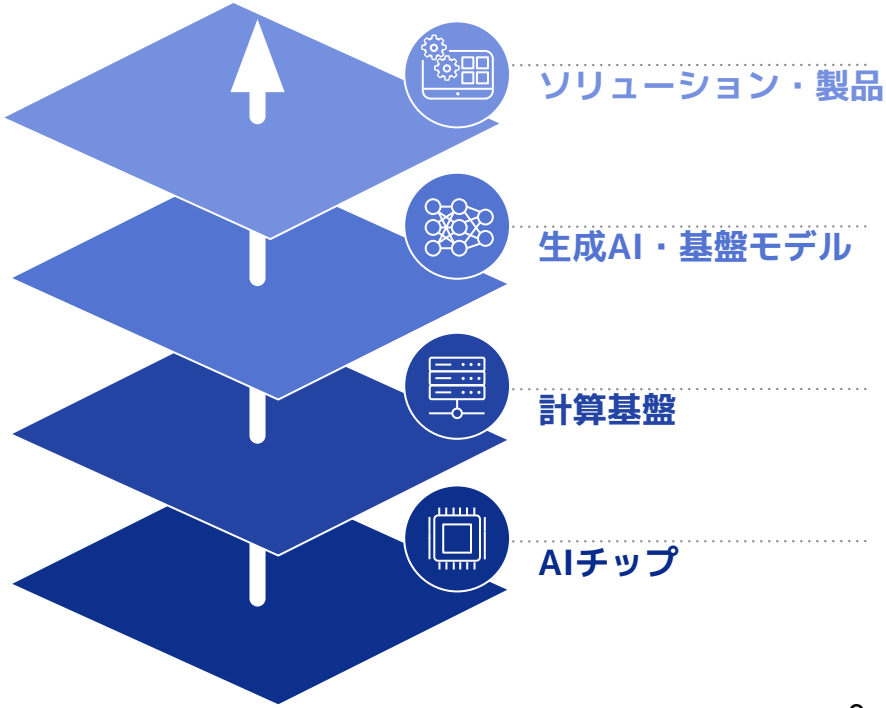
設立	2014年3月26日
本社	東京都千代田区
代表取締役	西川徹（最高経営責任者）、岡野原大輔（最高研究責任者）
従業員数	約300名（2024年1月）
事業内容	AIチップ、計算基盤、生成AI・基盤モデルなどのAI関連技術を活用したソリューション・製品の開発・販売および研究開発
主要子会社	株式会社Preferred Computational Chemistry（2021年6月） 株式会社Preferred Robotics（2021年11月） 株式会社Preferred Elements（2023年11月）
出資企業	トヨタ自動車株式会社　ファナック株式会社 日本電信電話株式会社　ENEOS ホールディングス株式会社 中外製薬株式会社　株式会社博報堂DYホールディングス 株式会社日立製作所　三井物産株式会社　みずほ銀行株式会社 東京エレクトロン株式会社（2024年9月末現在）

https://www.preferred.jp

Preferred Networks

PFNの事業: AI技術のバリューチェーンを垂直統合

PFNは、チップ、計算基盤、生成AI・基盤モデル、ソリューション・製品まで、AI技術のバリューチェーンを垂直統合し、ソフトウェアとハードウェアを高度に融合することで、競争力の高い技術の開発および産業応用を進めています。



様々な産業・消費者向けのソリューション・製品群

PreferredAI / Preferred Networks Visual Inspection / MATLANTIS / PFN 3D Scan / くいあこ rypko / kachaka

PLaMo

PLaMo Prime（国産LLM）
PLaMo Lite（エッジ向けSLM）

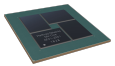
Preferred Potential (PFP)

物質の電子状態・
エネルギー計算モデル

GPUクラスタ

MN-3
(MN-Core™
クラスタ)

MN-Core™ 2のクラウド提供
（2024年開始予定）



MN-Core™

MN-Core™ 2

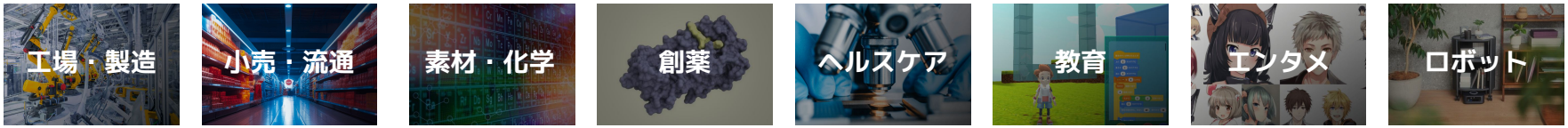
LLM向け
推論チップ
（2026年提供予定）

MN-Core
第三世代

Preferred Networks

PFNの事業: AI技術の水平展開

PFNは、AI技術のバリューチェーンを垂直統合し、産業、コンシューマー、社会に向けて様々な領域でソリューション・製品を水平展開しています。



産業
生産性向上・品質改善
属人化回避・人手不足解消

社会
安心・安全な社会
高度な教育・医療

消費者
人間の能力の拡張
新しい創作表現・娯楽体験

生成AI・基盤モデル

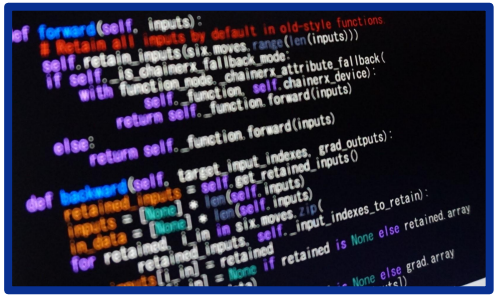
計算基盤

AIチップ

Preferred Networks

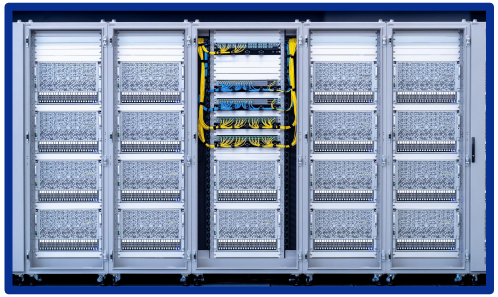
PFNの強み

AI関連技術・人材



- NeurIPS、AAAI、ICRAなど世界トップのAI・機械学習関連学会での論文採用実績
- Kaggle*称号保持者、ICPC*世界大会出場経験者が多数在籍
- 深層学習フレームワークChainer™を2015年に開発、現在はPyTorchの主要コントリビュータ

計算資源



- 大規模なスーパーコンピュータを自社運用
- AI向けプロセッサーMN-Core™シリーズを神戸大と共同開発
- 自社開発のスパコンMN-3がGreen500*で電力効率世界1位を3度獲得

ドメイン知識

- 産業ロボット、自動車、製薬、エネルギー等のリーディング企業との共同研究実績
- PFN Values（行動規範）に掲げる「死ぬ気で学べ」の精神で各業界ドメイン知識を学ぶ姿勢
- 様々なドメインの専門家が在籍

*Kaggle: 世界中の機械学習・データサイエンスに関わる人々のコミュニティーで、企業・政府などが提示した課題に対して最も精度の高いモデルの開発を競う世界的なコンペティション
*ICPC: 国際大学対抗プログラミングコンテスト
*Green500: 世界のスーパーコンピュータの性能上位500(TOP500)のうち、電力効率を評価するランキング

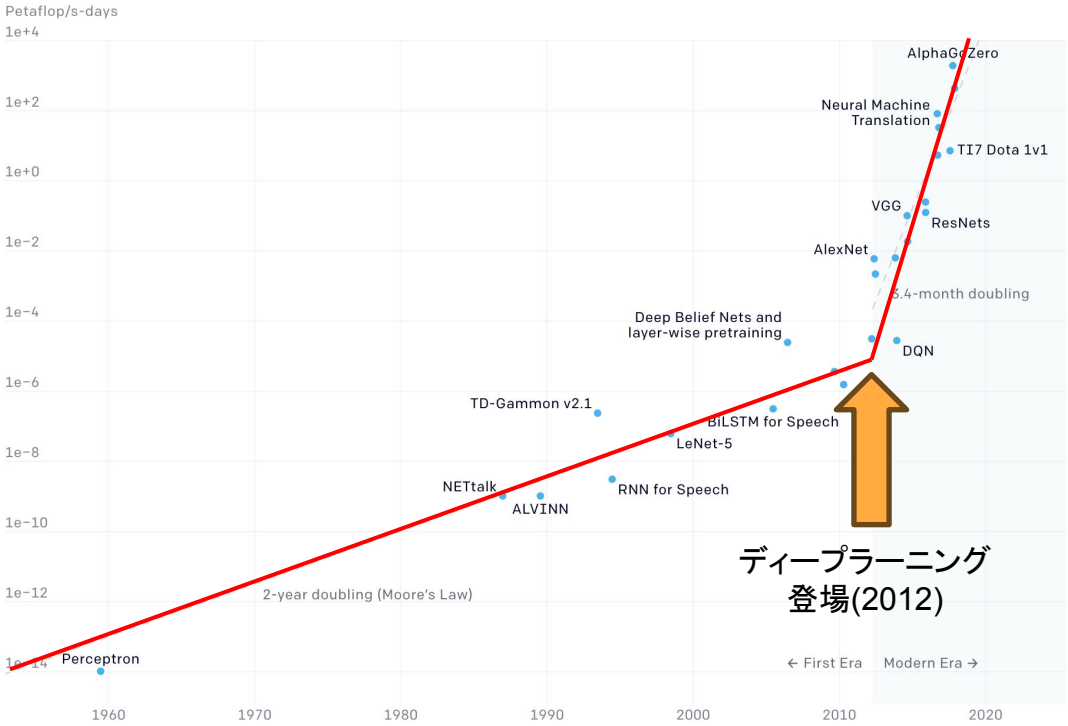
Preferred Networks

Preferred Networks

AIアクセラレータの歩み

AIに必要なリソース

- AIに必要な計算リソースは 3-4ヶ月ごとに 倍増



Preferred Networks

学習にかかるコストは膨大に

学習には **膨大な計算コスト** がかかる

GPT-3(自然言語処理)の学習に要した計算コスト

約 **1000** 台
GPU V100

約 **数百** 日
学習時間

約 **12** 億円
計算コスト

Llama 3.1(405B)の学習に要した計算コスト

約 **16000**
台
GPU H100

約 **百** 日
学習時間

約 **9200** 億
円
計算コスト

($60Mと試算)

一度学習すれば
数十の自然言語処理タスクに再利用可能に

同様のトレンドは画像認識、音声認識、機械翻訳、自動運転、材料探索、創薬、ゲノム解析などでもみられている

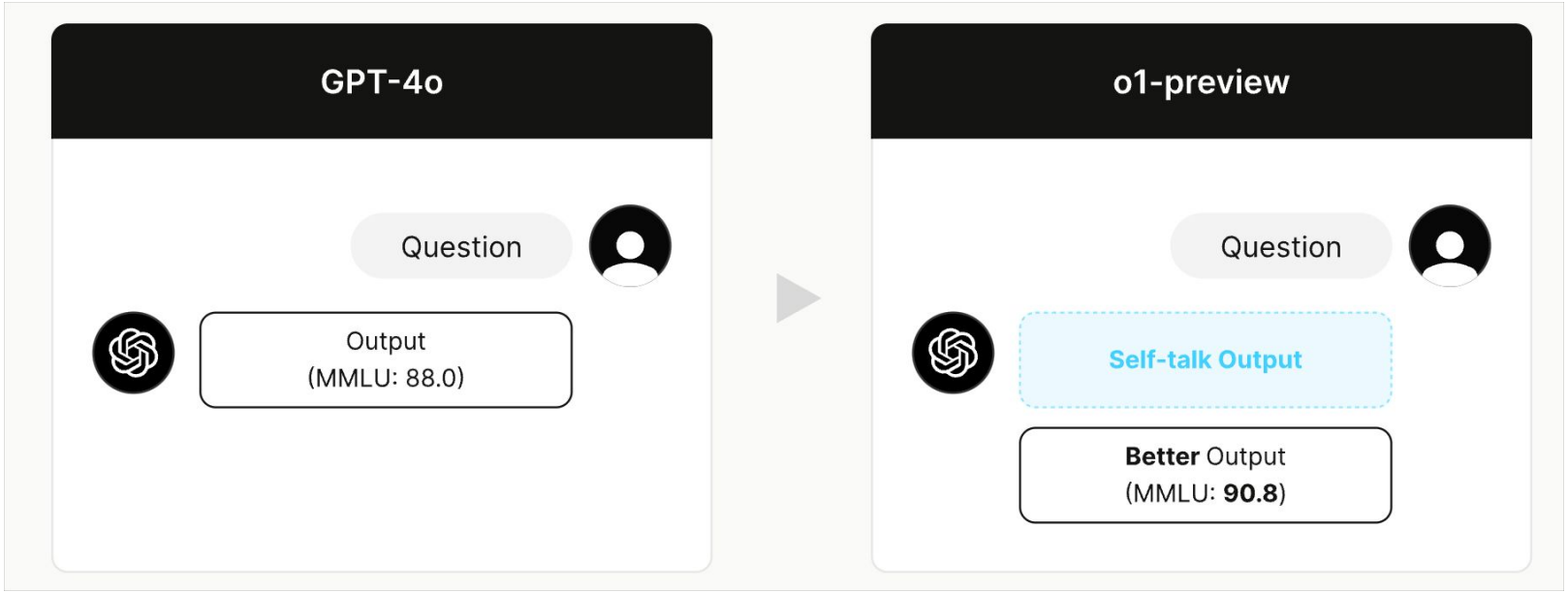
膨大な学習コストをかけてでも巨大な基盤モデルを作ることのメリットがある！

推論時スケーリング則：長く、深く考える

ユーザからは隠された、自身のみが見える「隠れトークン」を生成

人間がメモを取りながら考えるように、内部での思考、振り返りを行う

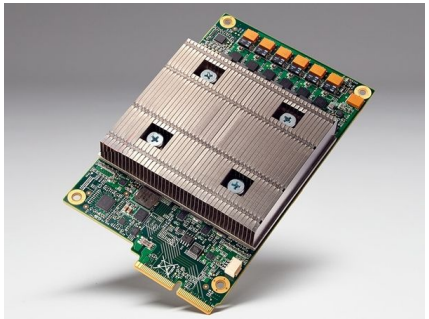
=> 性能が向上した一方推論コストも大幅に上昇



Preferred Networks

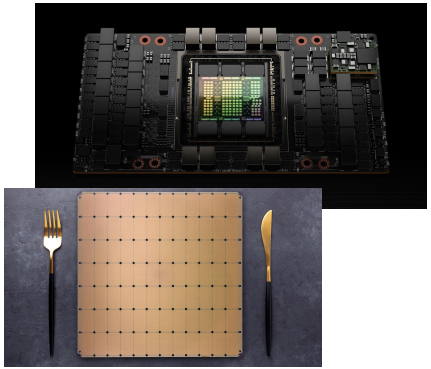
AIを高速に処理しよう！

ディープラーニング黎明期
Alexnetを初めとした、
Convolutional Neural
Networkを支えたNVIDIA
GPU



CNN 全盛期
専用アクセラレータによる
様々な社会実験 …

Google TPUなど



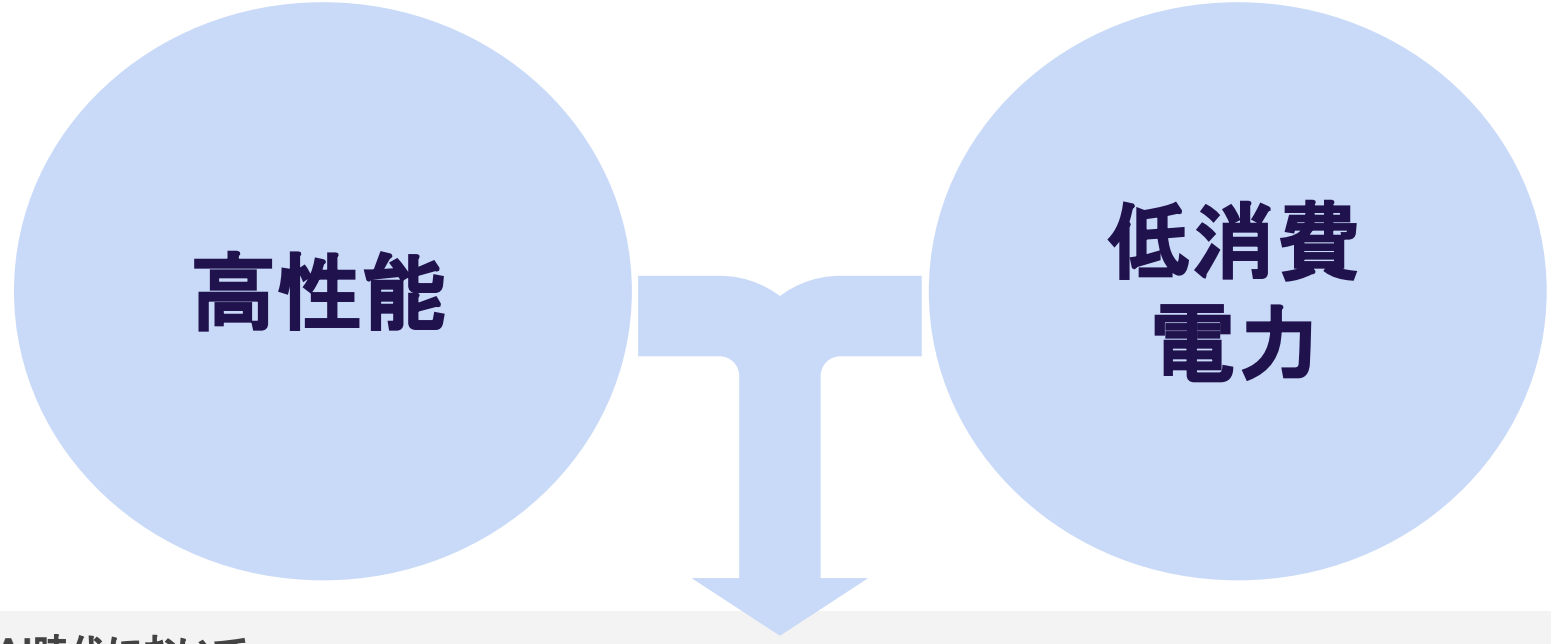
生成AI全盛期
学習と推論で大きな性能要
件の差が発生

NVIDIA H100
Cerebras WSE-3
SambaNova SN40

2012 2016 2023

Preferred Networks

AI半導体に求められる要素



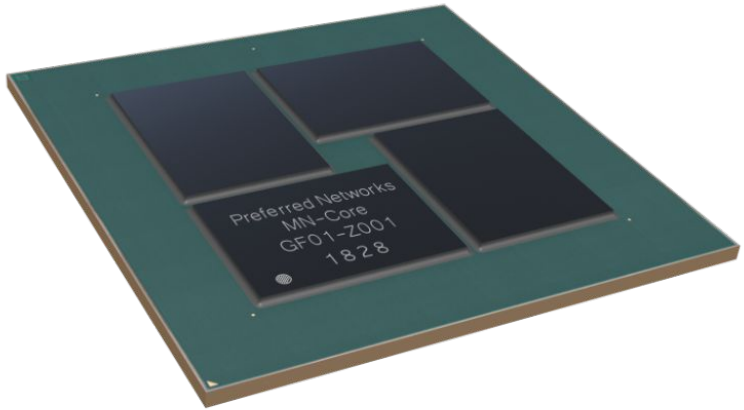
生成AI時代において
計算インフラに求められるニーズ

コスト

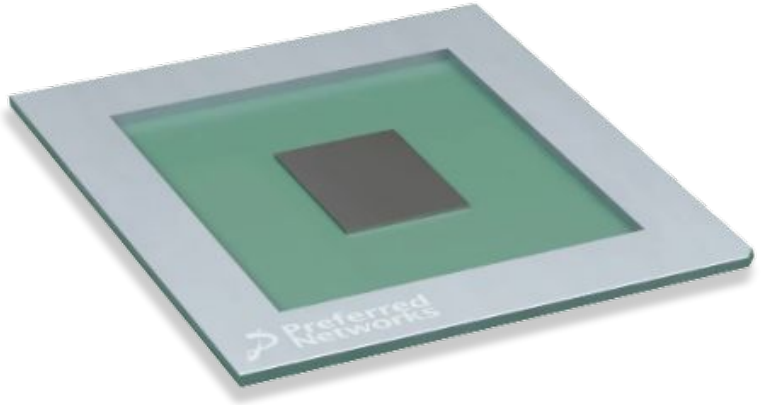
計算量

GHG排出削減

MN-Core series



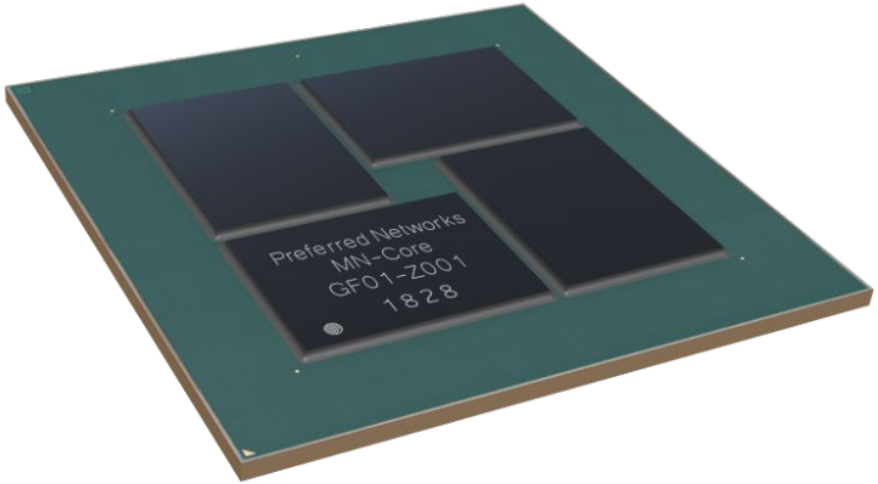
MN-Core



MN-Core 2

MN-Core™ Series

Preferred Networks



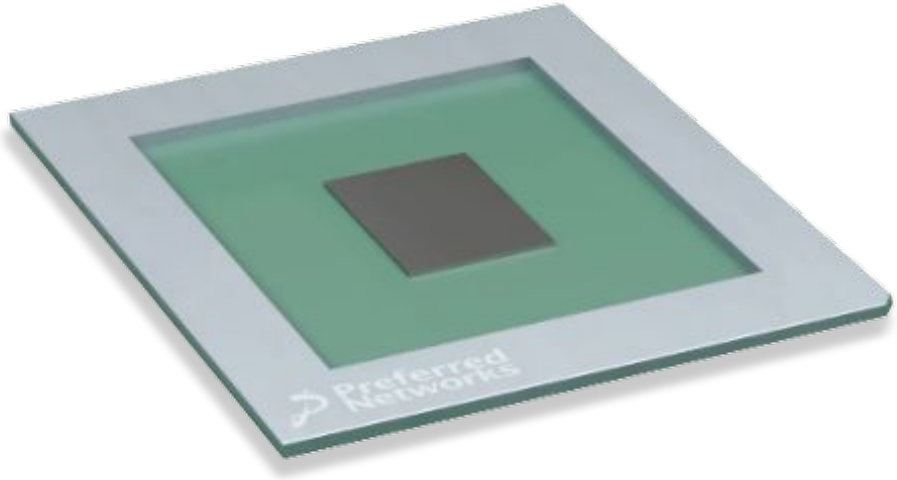
MN-Core

第一世代 MN-Core

- TSMC 12nm
- 500MHz / 500W

	Flops	GFlops/W
倍精度	32.8 T	66
単精度	131 T	260
半精度	524 T	1000

MN-Core 2



MN-Core 2

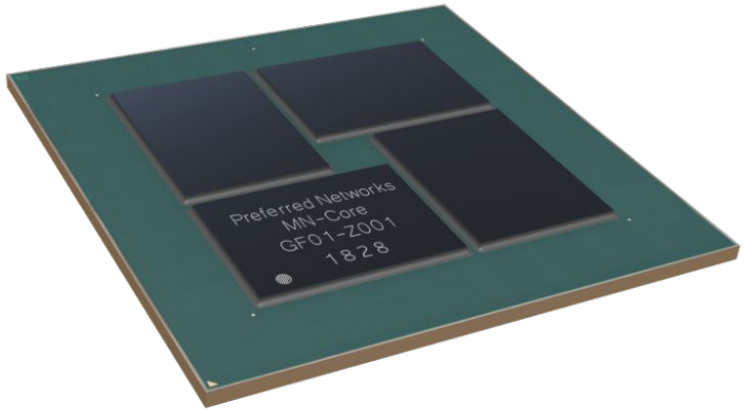
第二世代 MN-Core

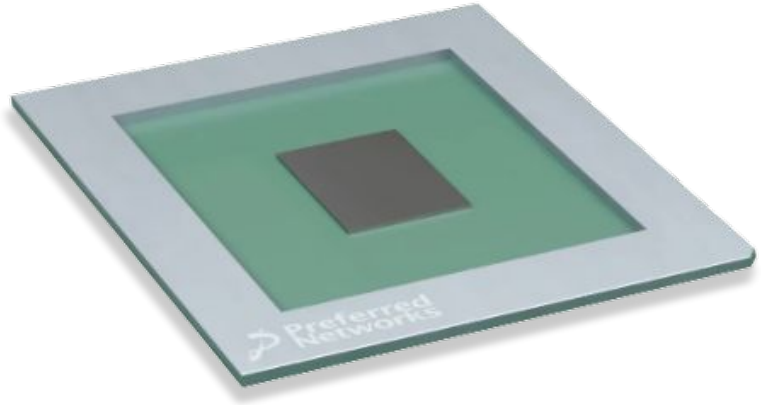
- TSMC 7nm
- 750MHz / 322W

	Flops	GFlops/W
倍精度	12 T	37.24
単精度	49 T	148.9
疑似単精度	98 T	297.9
半精度	393 T	1192

Preferred Networks

MN-Core シリーズの位置付け





MN-Core
実証実験モデル

実際にどういった用途で使えるのか
どういったモデルが動くのかを実証した

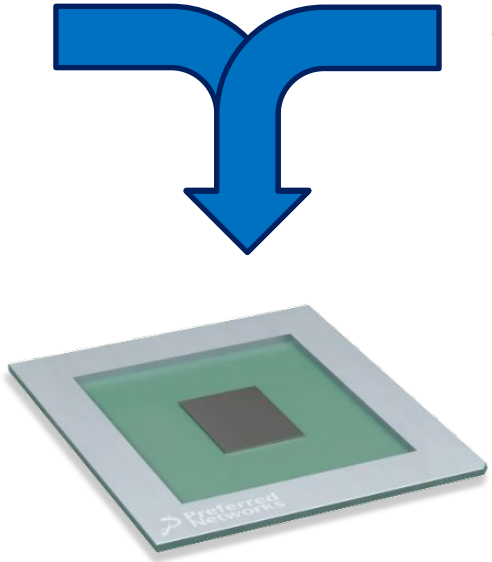
MN-Core 2
実験 + 普及モデル

MN-Coreを取り回しが良くなるように
修正・改良。実ワークロードへの適用
を目指す

MN-Coreの紹介

- MN-Coreの特徴

消費電力性能、
高いシリコン効率を意図した
チップ設計

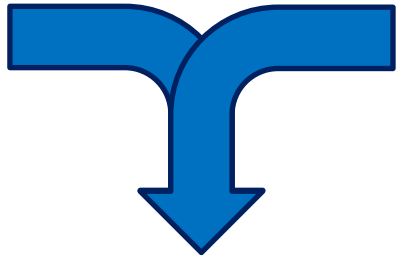


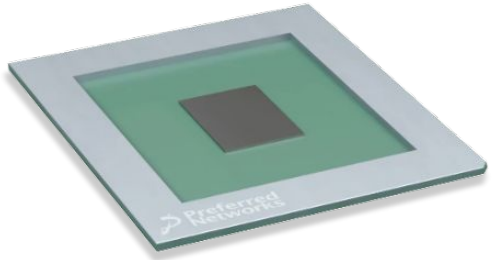
ソフトウェアによる高度な最適化を可能にするアーキテクチャ

MN-Coreの紹介

- MN-Coreの特徴

消費電力性能、
高いシリコン効率を意図した
チップ設計

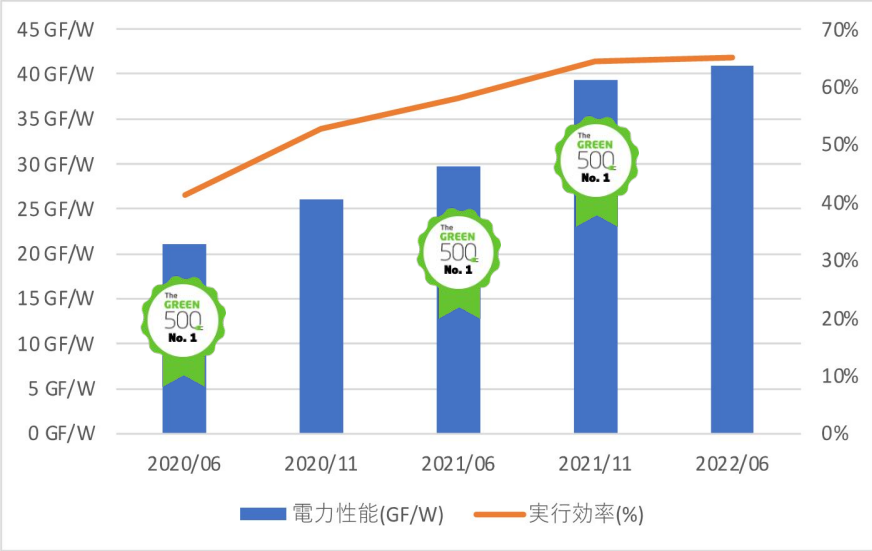


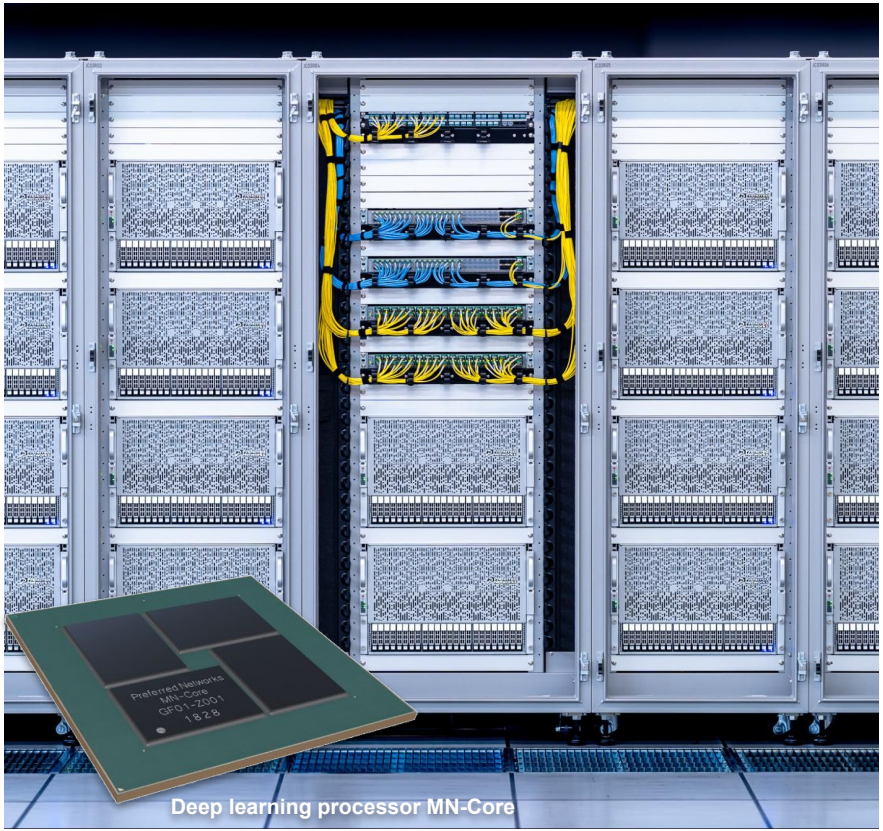


ソフトウェアによる高度な最適化を可能にするアーキテクチャ

Preferred Networks

MN-Core Accelerates World's Most Efficient Supercomputer



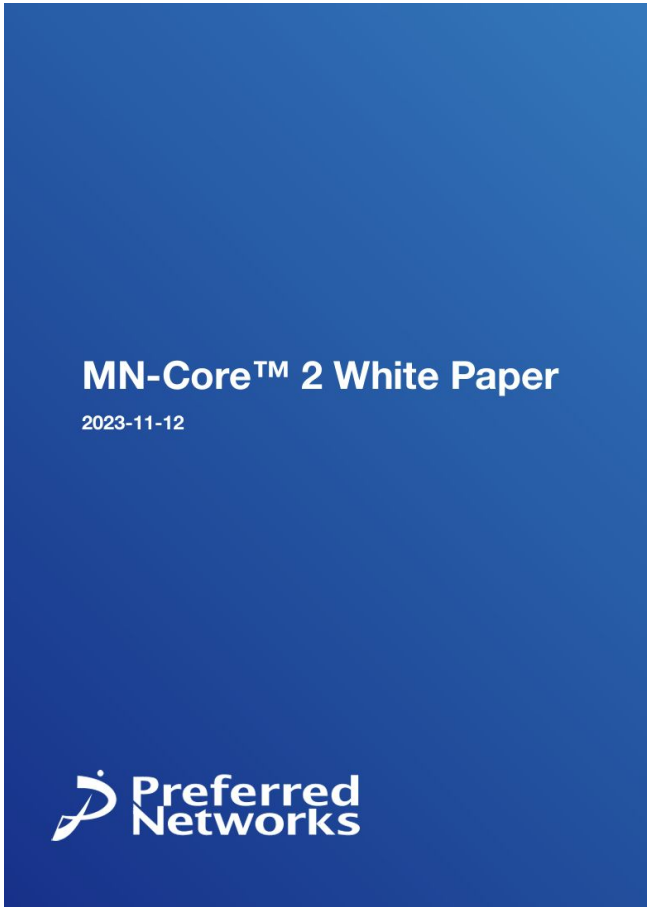


Our MN-3 supercomputer is certified as the world's most efficient supercomputer (Green500 List Nov. 2021).

Important effort towards achieving carbon neutral AI datacenter

Preferred Networks

MN-Core 関連資料



MN-Core™ 2 ソフトウェア開発者マニュアル

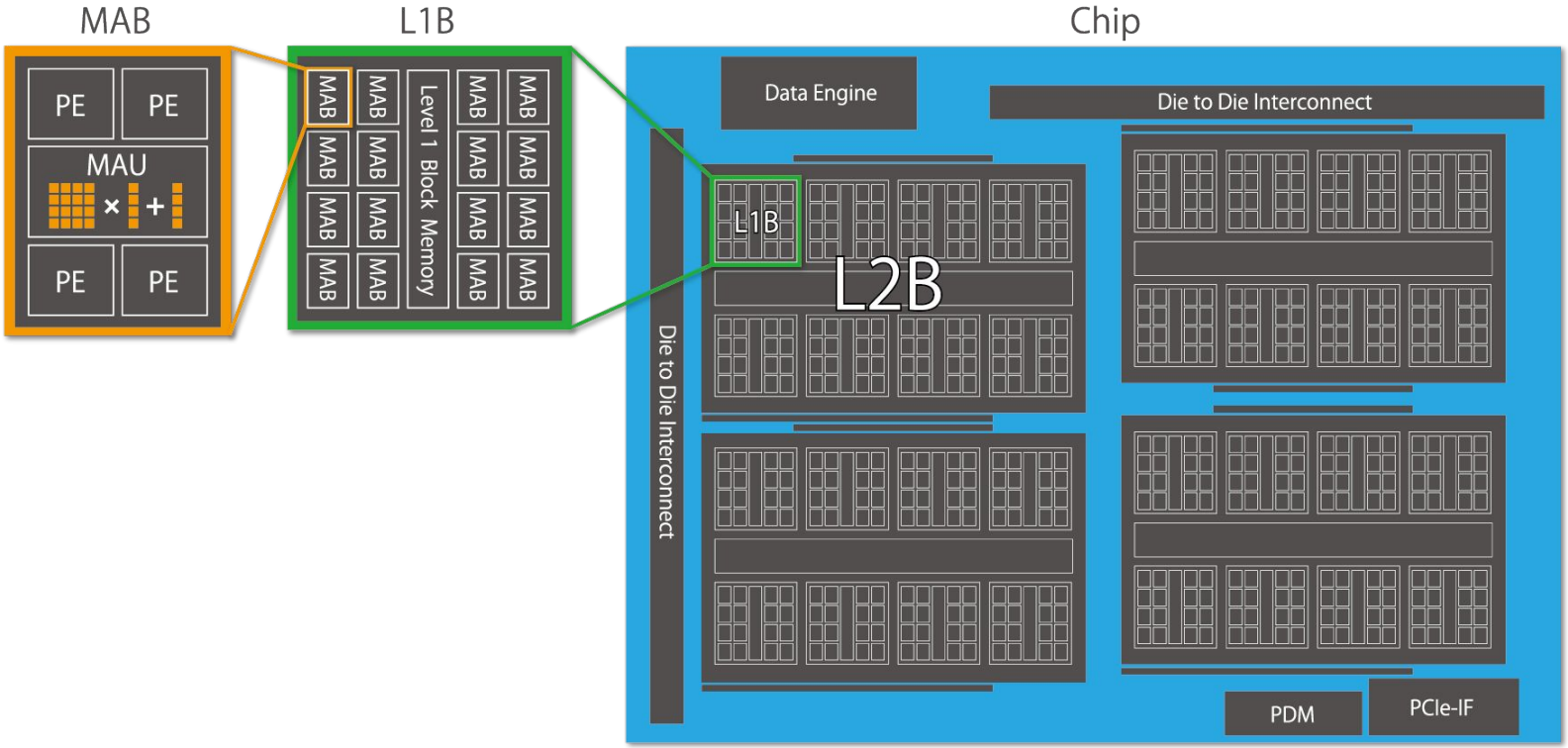
株式会社 Preferred Networks

2024 年 2 月 21 日

Preferred Networks

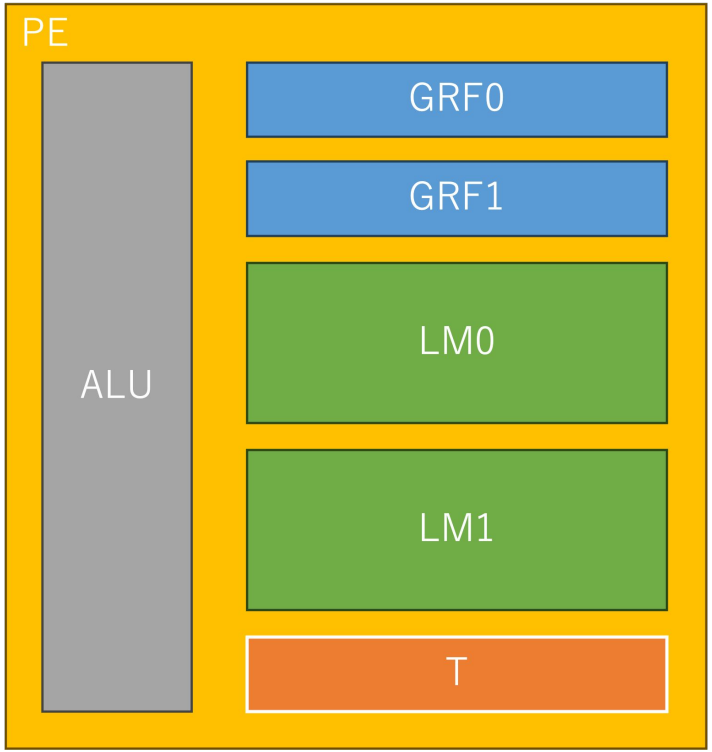
MN-Core概要

- MN-Core series Architecture Blockdiagram



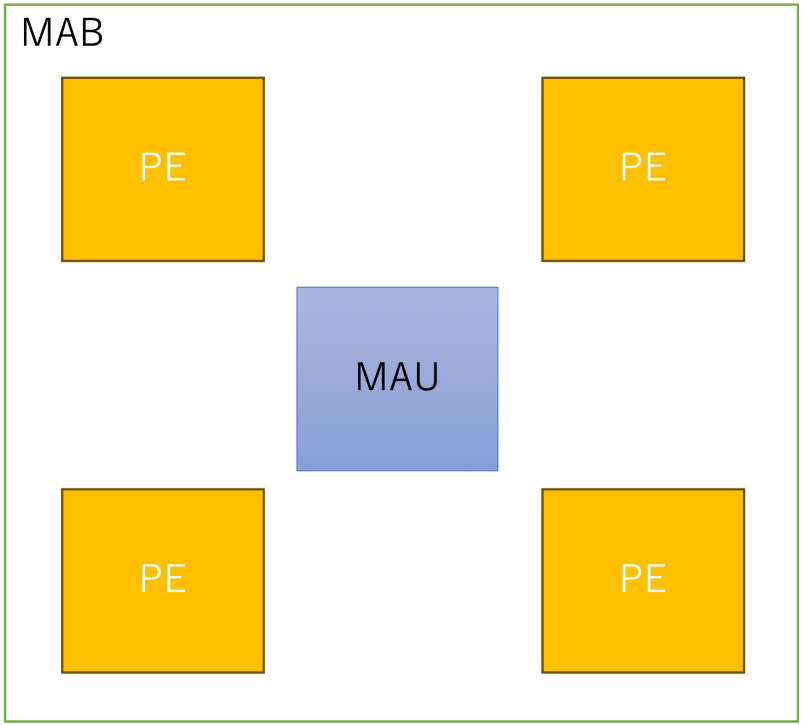
Preferred Networks

PE



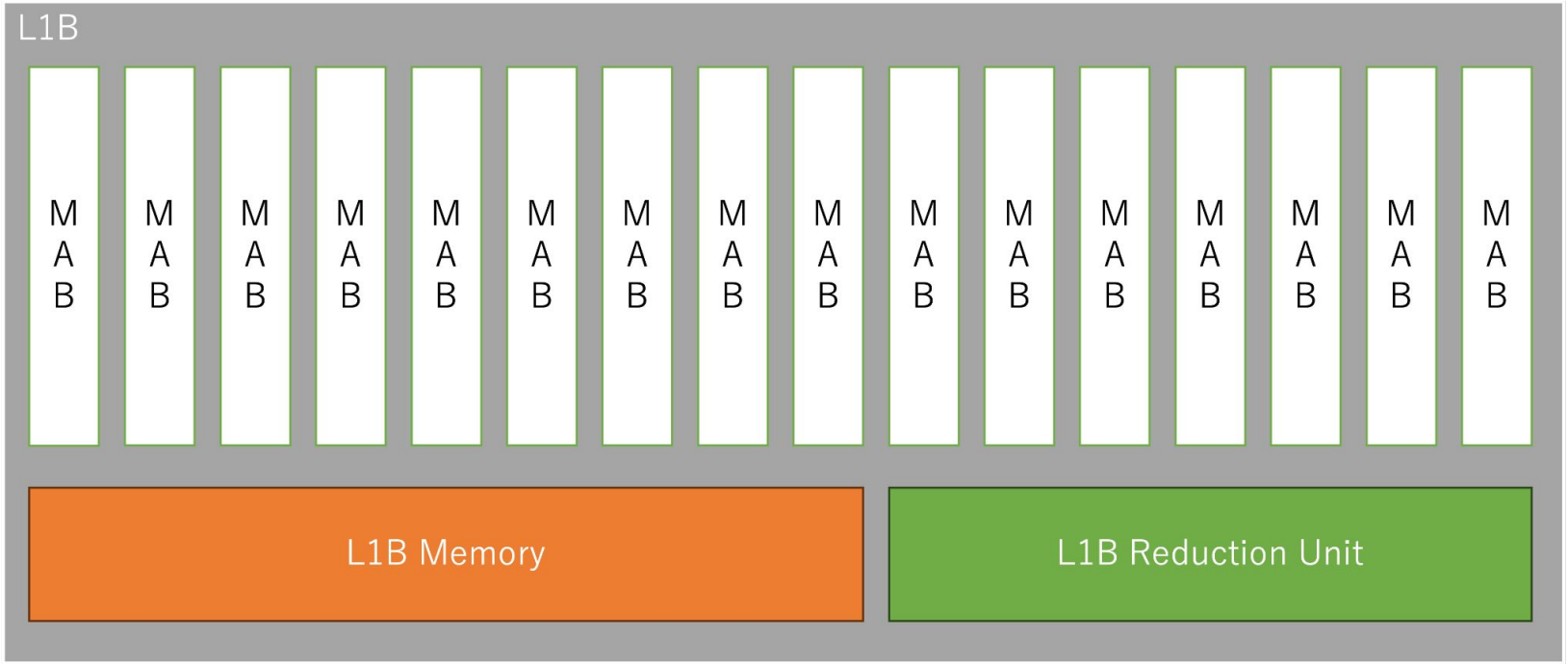
- PE
  - 実際に演算を担当するコア部分
  - ALUでは整数演算を処理可能
    - 特殊な命令ではReLUやBlock Float変換など
  - PE内 64bit SIMDもある
    - double: 64x1
    - float: 32x2
    - half: 16x4
  - **たくさんの SRAMにより構成されている**

MAB



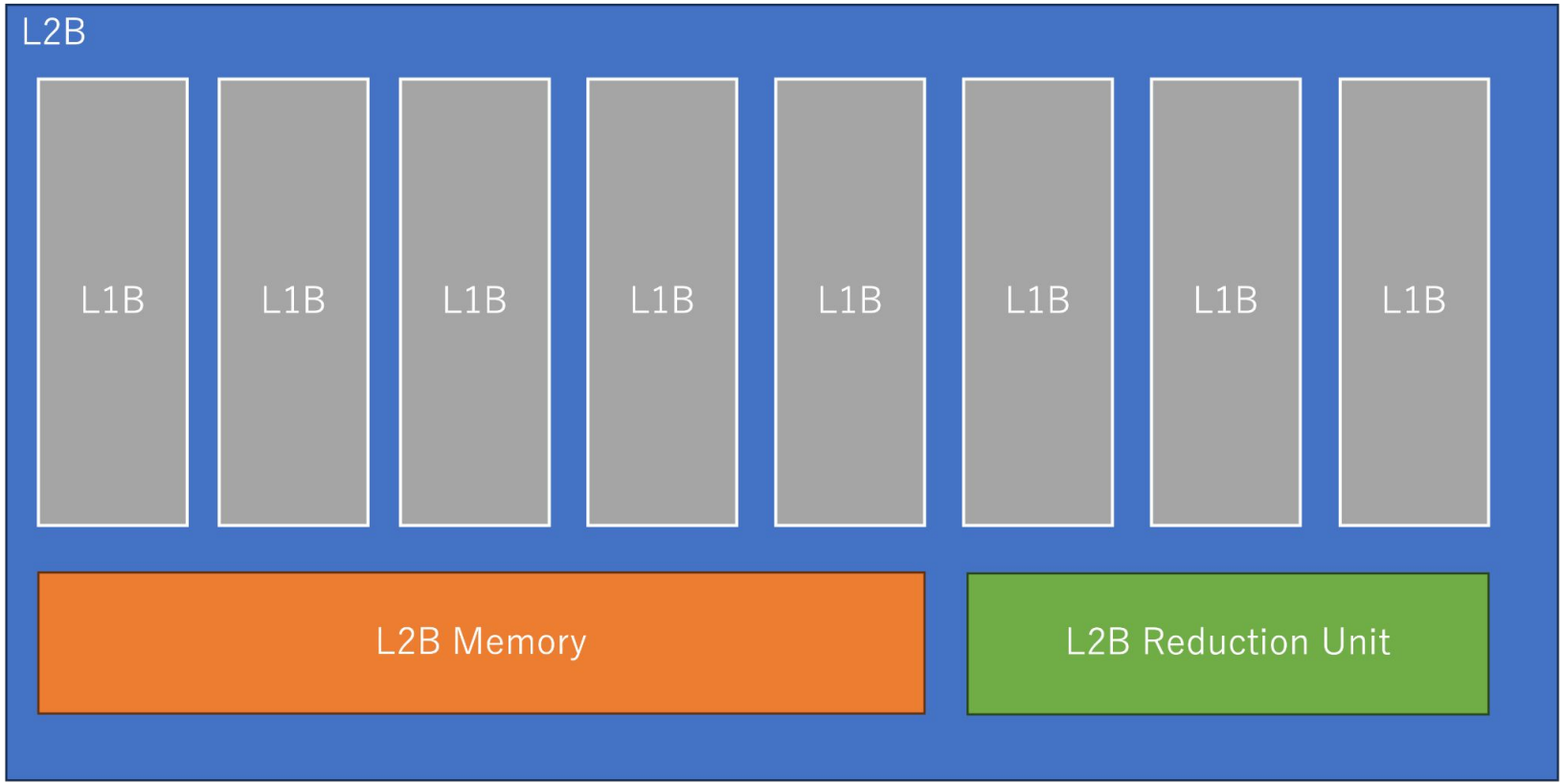
- MAB(Matrix Arithmetic Block)
  - 4つのPEと、1つのMAU(Matrix Arithmetic Unit)で構成されている。
  - MAUはPEから入力データを受け取り、浮動小数点演算を行い、PEに戻す。
  - MAUは行列演算用の行列レジスタを持っており、行列ベクトル積で使うための行列を保持する。
    - 行列レジスタは2面あるので、行列のdouble-bufferingが可能
- MAUでできる演算
  - 積和演算
    - ベクトル積和(A x B + C) :半精度、単精度、倍精度
    - 行列積和(A x 行列レジスタ+ C) :半精度、疑似単精度、単精度、倍精度
  - 行列レジスタ書き込み:半精度、疑似単精度、単精度、倍精度
  - 行列レジスタ転置読み出し:半精度、疑似単精度、単精度、倍精度

L1B



Preferred Networks

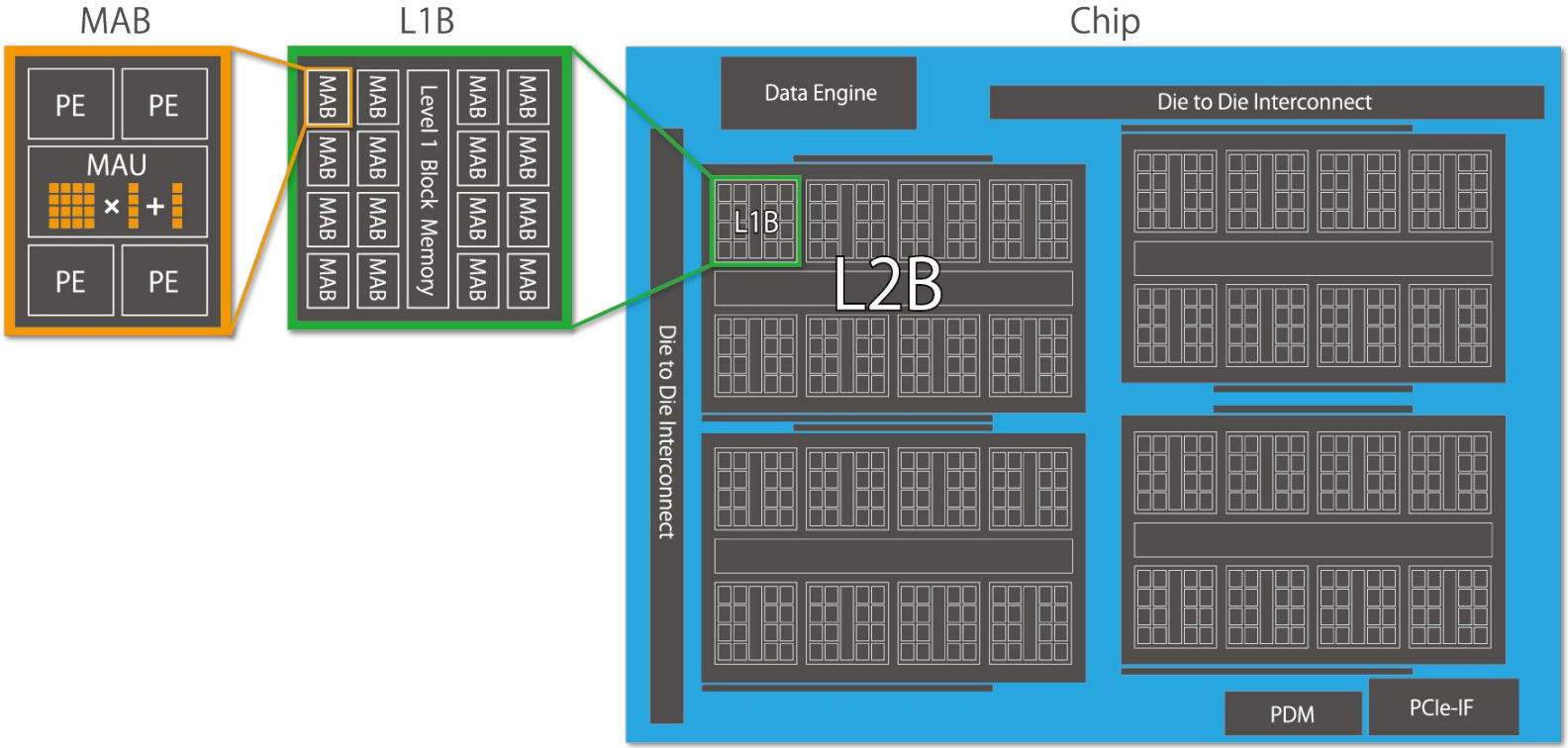
L2B



Preferred Networks

MN-Core概要

- MN-Core series Architecture Blockdiagram

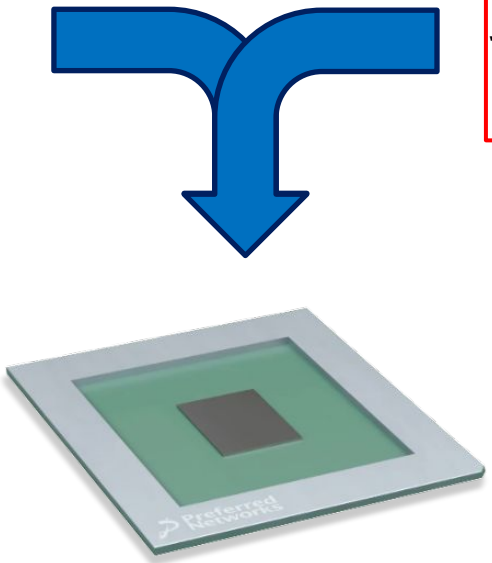


Preferred Networks

MN-Coreの紹介

- MN-Coreの特徴

消費電力性能、
高いシリコン効率を意図した
チップ設計



ソフトウェアによる高度な最適化を可能にするアーキテクチャ

Preferred Networks

ソフトウェアに対する高度な最適化を可能にするアーキテクチャ

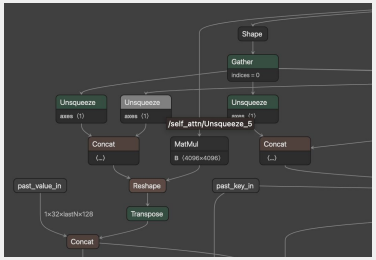
AIワークロードはこれまでの汎用計算と異なる特性を持つ

これまでの汎用計算

```
for(int y = 0;y < ih;y++){
    for(int x = 0;x < iw;x++){
        auto acc = 0;
        for(int yk = 0;yk < kh;yk++){
            for(int xk = 0;xk < kw;xk++){
                acc += image[y+yk][x+xk] *
                    filter[yk][xk];
            }
        }
        image[y][x] = acc / (kw * kh);
    }
}
```

ワークロードごとにオーダーメイドで定義される、条件分岐や繰り返しで定義される**動的な手続き**

AIに必要とされる計算



AIモデルとして標準化された、計算の依存関係が明にグラフとして表現された**静的な手続き**

既存の汎用プロセッサー

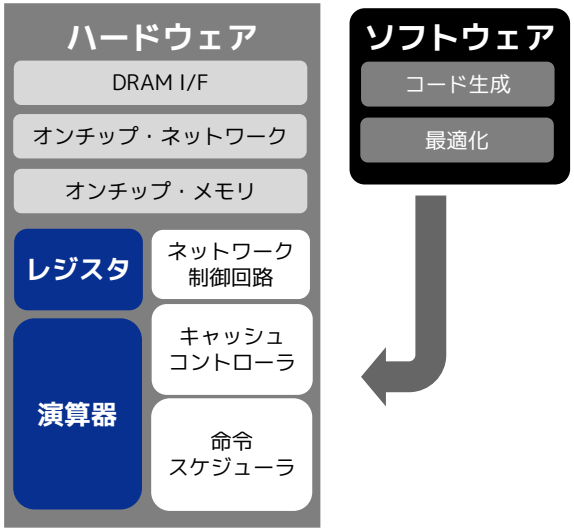
実際の演算を行う浮動小数点演算部は小なく、多くの面積は動的な手続きを効率よく動かすために利用されている(キャッシュ/投機実行など)

静的な手続きを前提としたアーキテクチャ革新の可能性

Preferred Networks

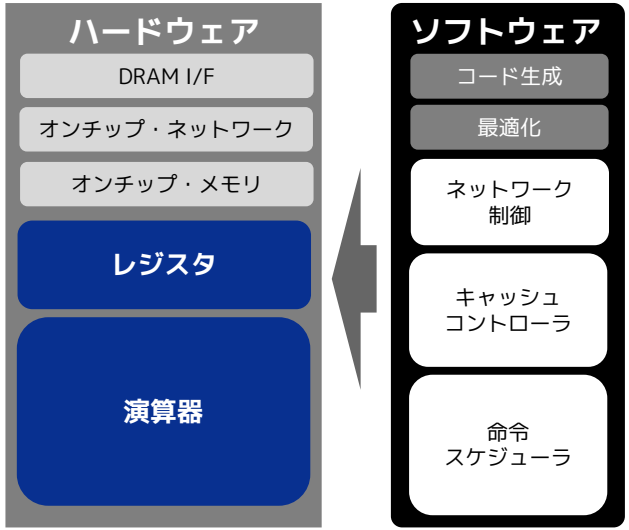
MN-Core 概要

一般的な計算機



汎用プロセッサー

MN-Core Architecture

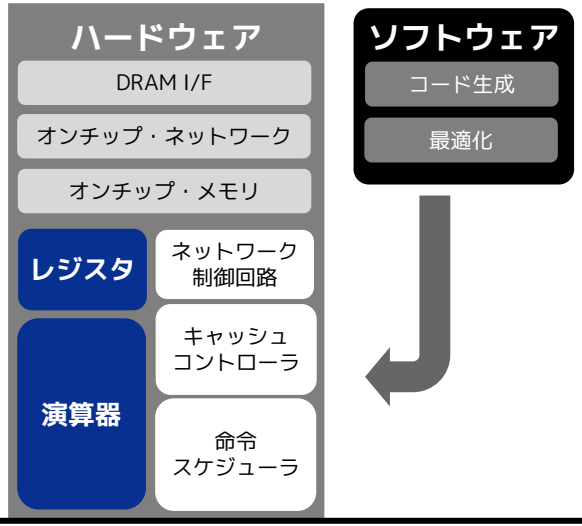


MN-Coreシリーズ

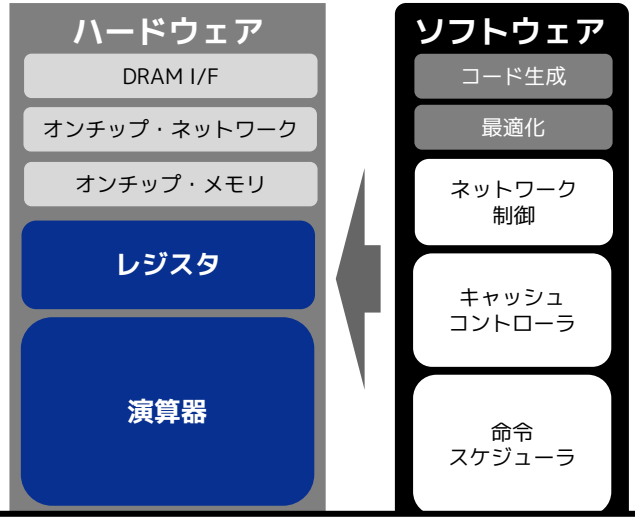
Preferred Networks

MN-Core 概要

一般的な計算機



MN-Core Architecture



ハードウェアで制御する代わりにソフトウェアで制御することにより、ハードウェアの効率化を図るアーキテクチャ

Preferred Networks

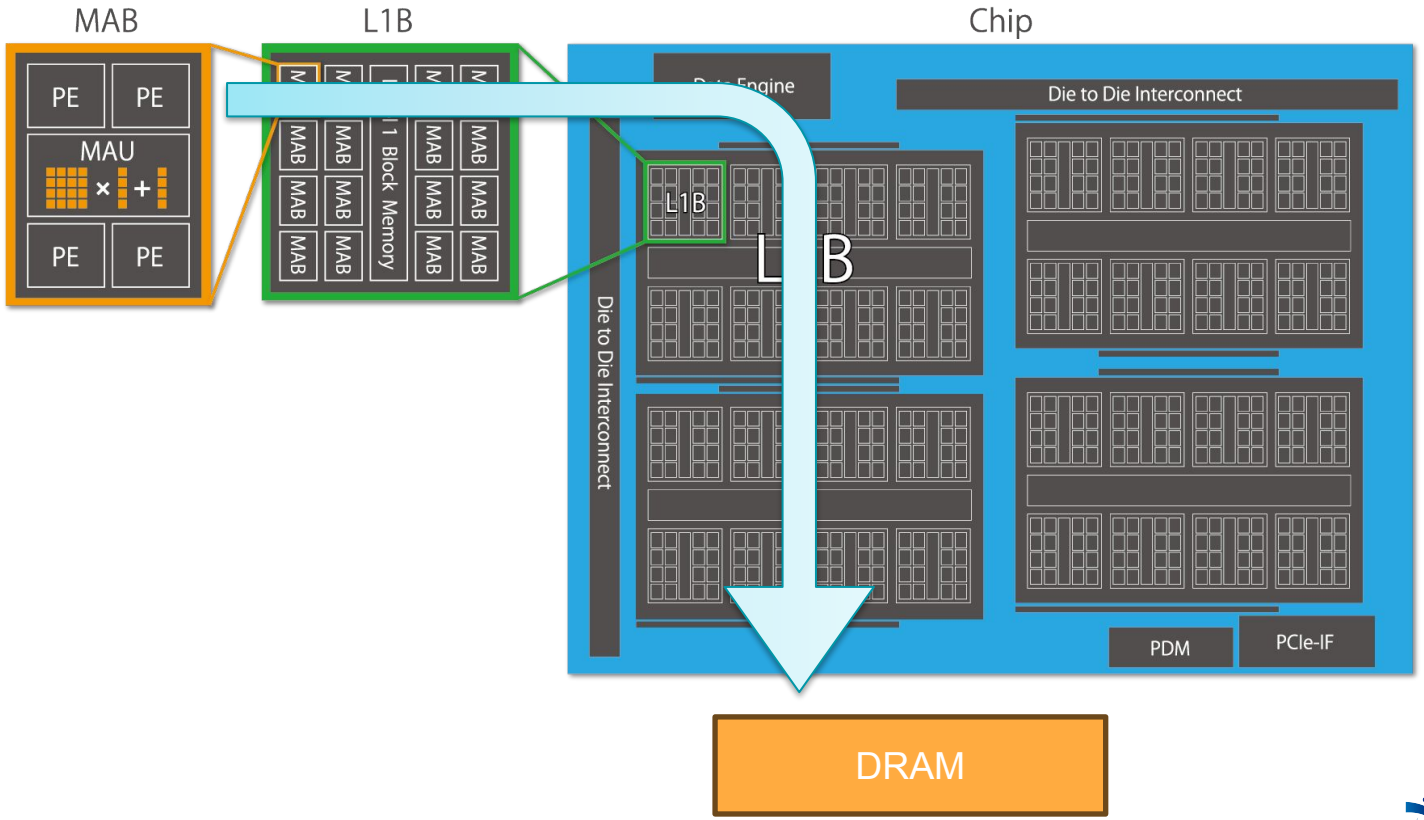
2つの重要ポイント

1. メモリ空間が独立したNear Memory Architecture
2. ハードウェアとしての制御機構を持っていない

2つの重要ポイント

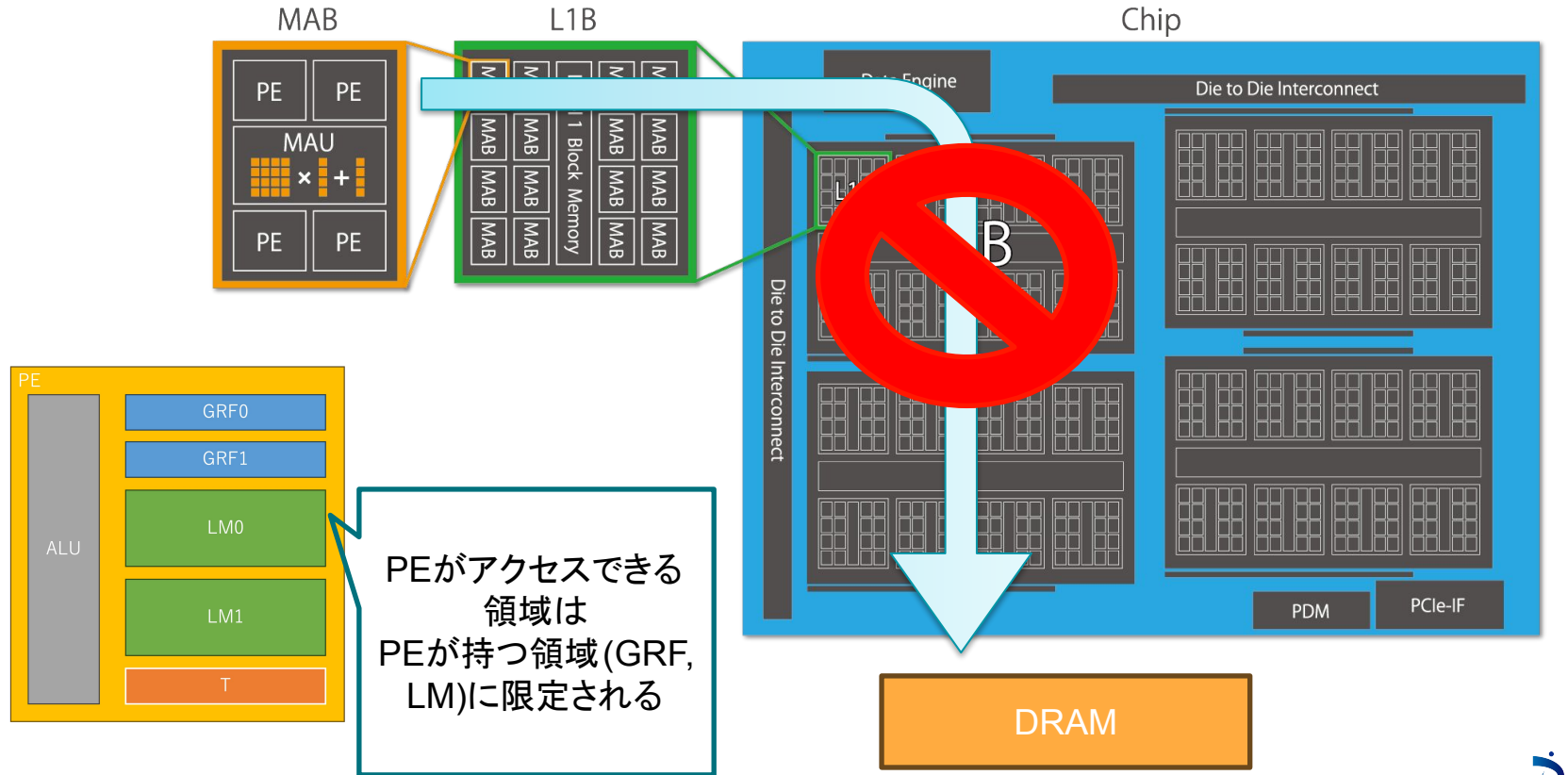
1. メモリ空間が独立したNear Memory Architecture
2. ハードウェアとしての制御機構を持っていない

メモリ空間が独立しているということ



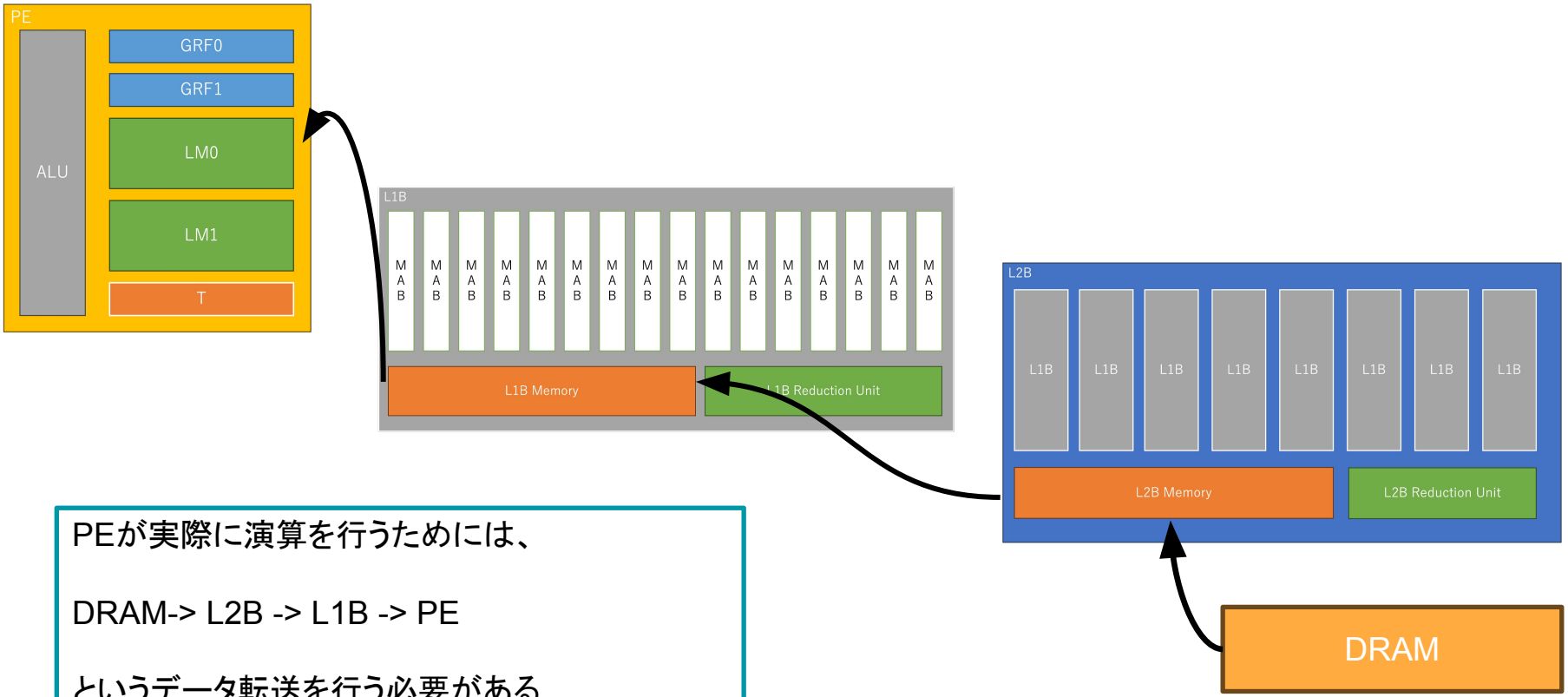
Preferred Networks

メモリ空間が独立しているということ



Preferred Networks

メモリ空間が独立しているということ



PEが実際に演算を行うためには、

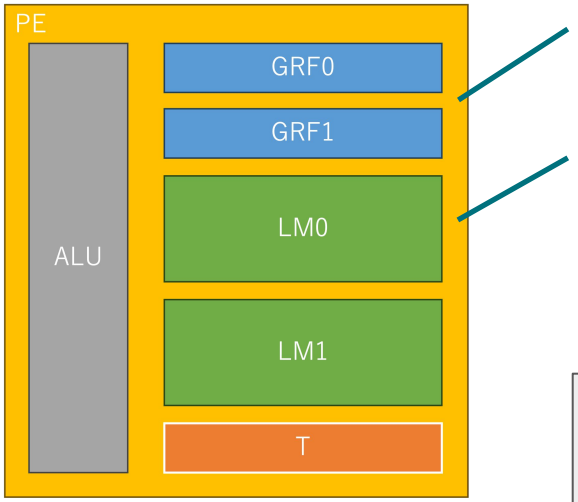
DRAM-> L2B -> L1B -> PE

というデータ転送を行う必要がある

Preferred Networks

Near-memory Computing Architecture

- 演算器近傍の高速なメモリの活用を前提にしたアーキテクチャ



GRF0, GRF1: 2KB/unit 1 read 1 write / cycle = 32Byte / cycle

LM0, LM1: 16KB/unit 1 read or 1 write / cycle = 16Byte / cycle

Preferred Networks

Near-memory Computing Architecture

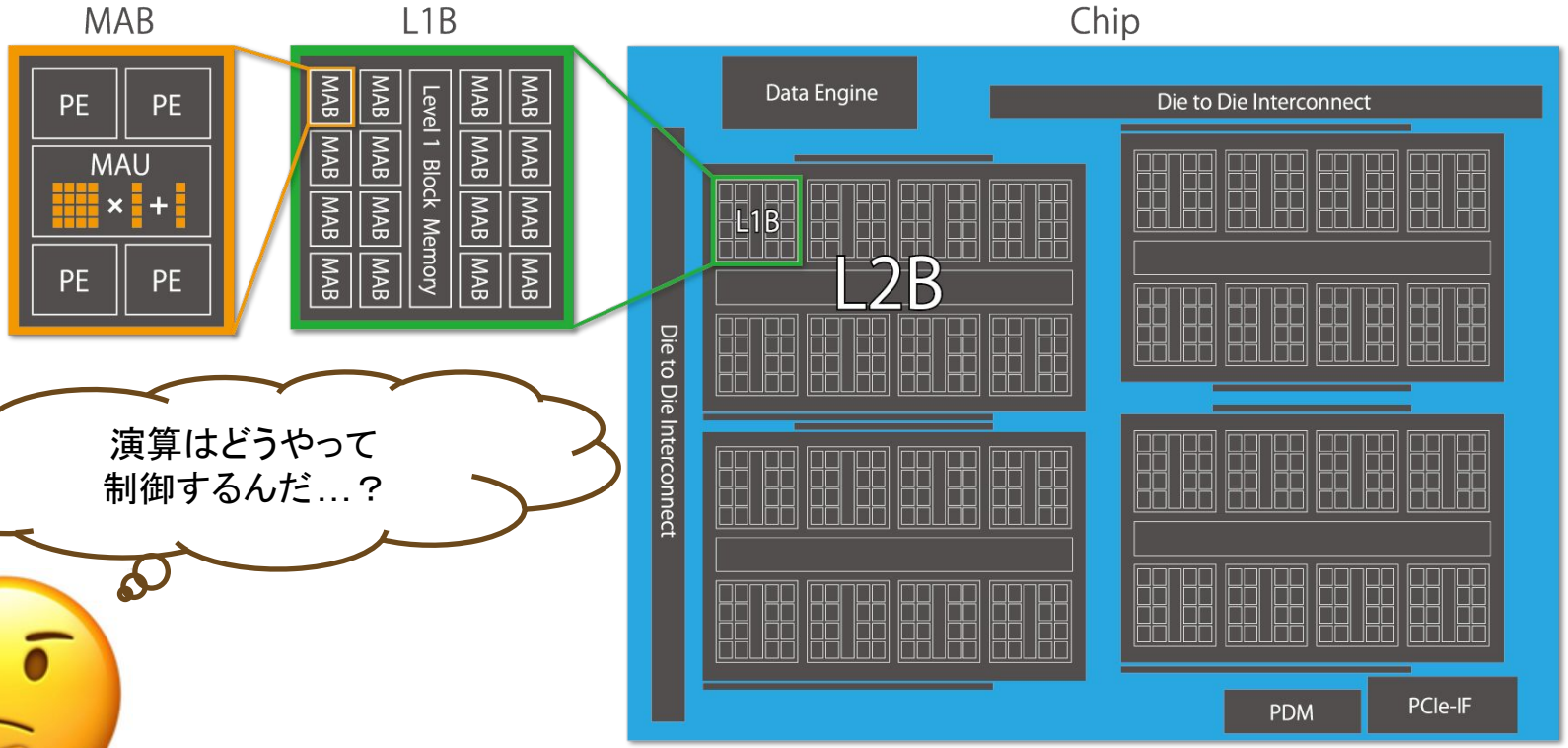
- 演算器近傍のメモリが巨大だと、計算にどういった影響がある？
  - DRAMへのアクセスを緩和することができる
    - メモリ帯域ネックのアプリケーションの性能を向上させうる可能性がある
    - Temporal Blocking
  - DRAMにデータを退避するのではなく、再計算してデータを再構成する
    - 再計算を用いたMN-Core向けコンパイラの最適化

2つの重要ポイント

1. メモリ空間が独立したNear Memory Architecture
2. ハードウェアとしての制御機構を持っていない

ハードウェアとしての制御機構を持っていない

- MN-Core series Architecture Blockdiagram

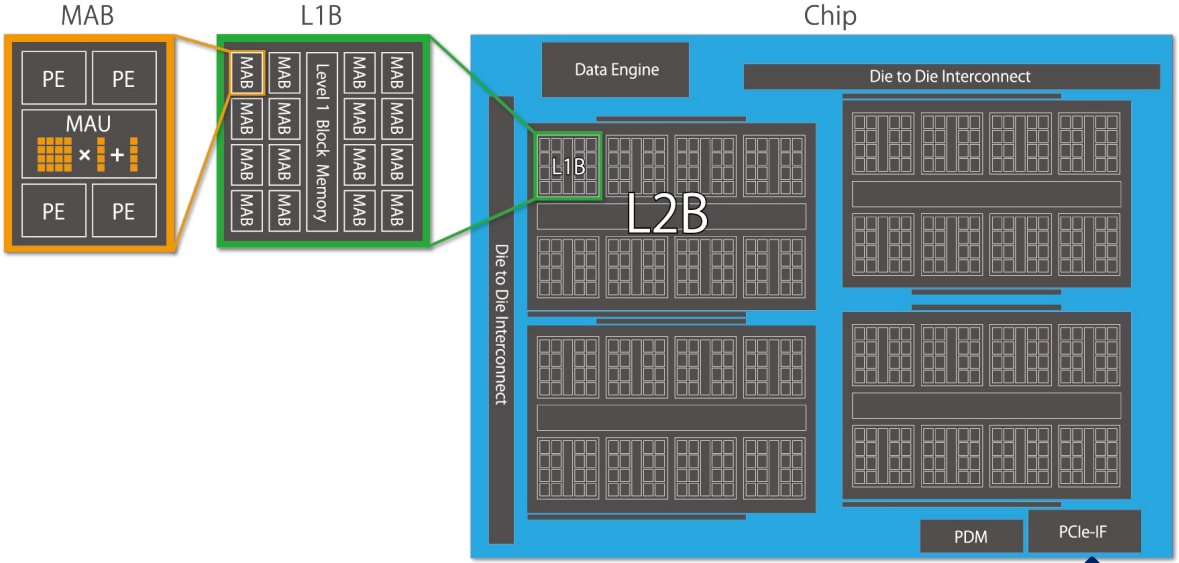


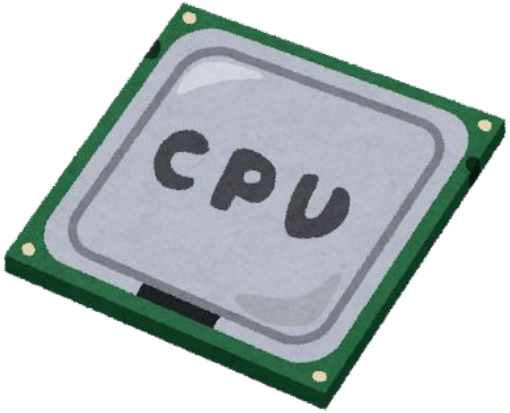
Preferred Networks

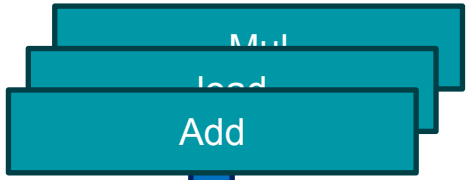
つまりどういうこと？

- MN-Coreは、PEに命令発行ユニットを持たない！
- 同様に、PEに制御ユニットも持たない！
  - プログラムカウンタとかもない！

- ではどのように演算を制御するの？







ホストプロセッサ上で命令列を生成し、PCIe経由で流し込む

ボード全体で単一の命令を実行する => **SIMD アクセラレータ**

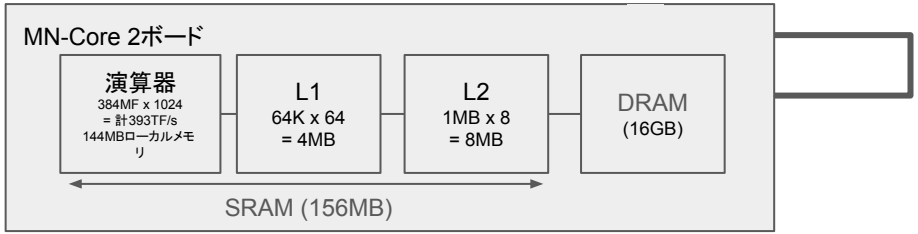
Preferred Networks

MN-Coreにおける命令とは？

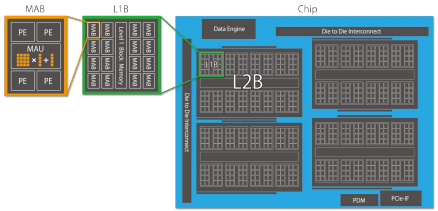
- MN-Coreにおける命令とは以下のように定義されている
  - PE命令
  - MAU命令
  - L1B 転送命令(L1B <=> MAB間転送命令)
  - L2B 転送命令(L2B <=> L1B間転送命令)
  - DRAM / PDM 転送命令(DRAM <=> L2B間転送命令)
- これらは、すべて同時に発行することができる

MN-Coreを使いこなすポイント

演算器近傍の高速なメモリにいかにデータを配置するか



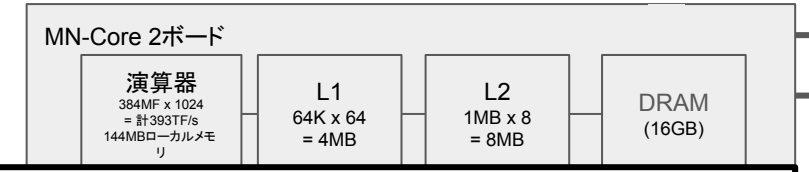
VLIW x SIMDの命令をいかに駆使するか



Preferred Networks

MN-Coreを使いこなすポイント

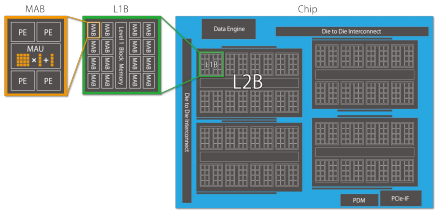
演算器近傍の高速なメモリにいかにデータを配置するか



VLIW

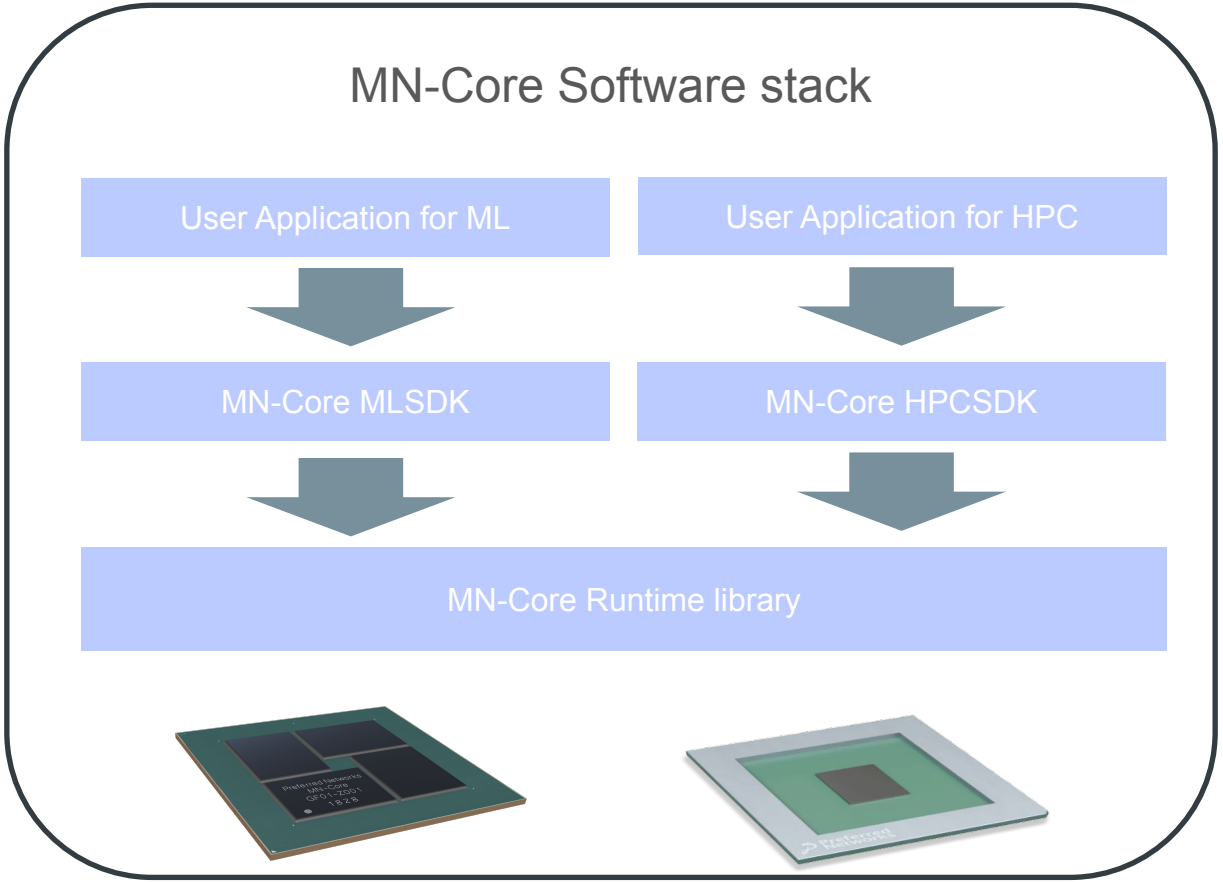
ハードウェアを最大限効率化するための
ソフトウェアスタックが非常に重要！

転送命令



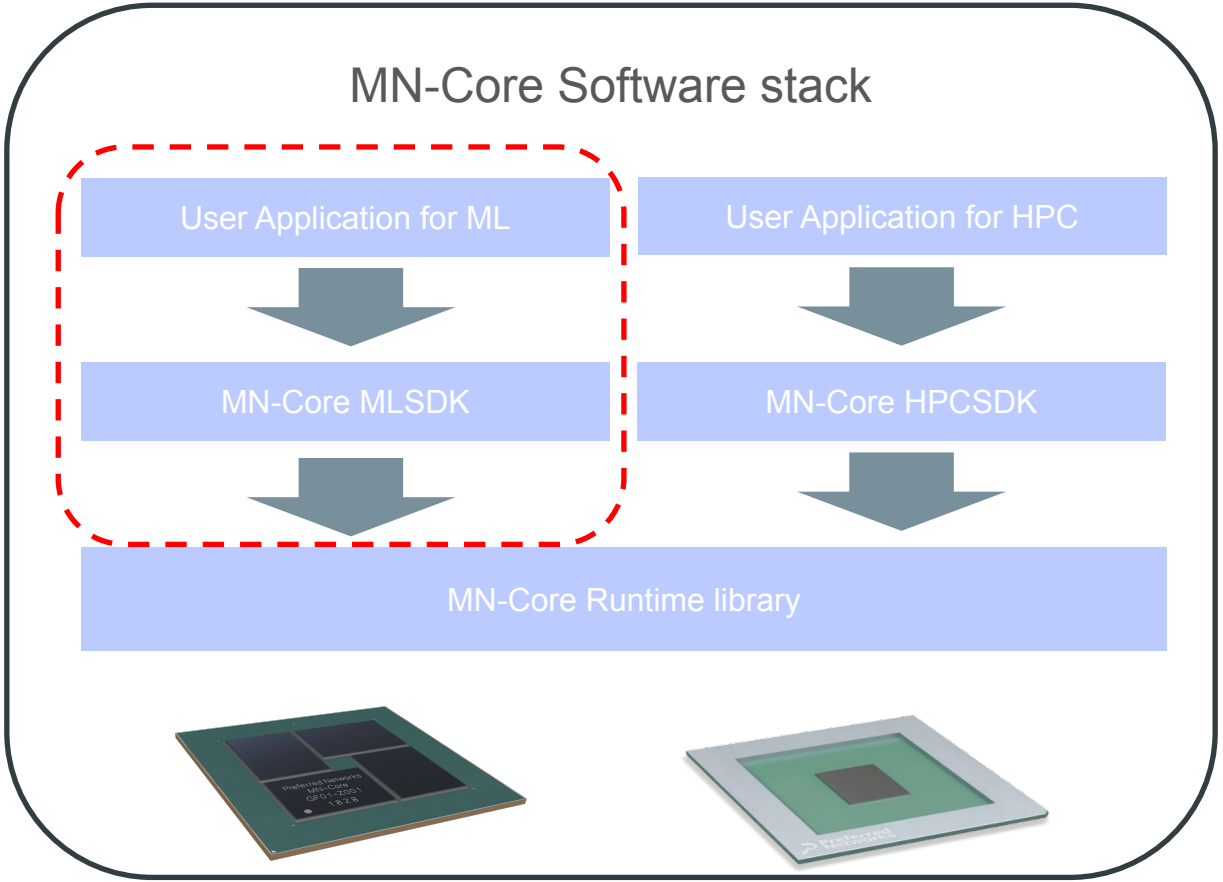
Preferred Networks

MN-Core Software Stack



Preferred Networks

MN-Core Software Stack



Preferred Networks

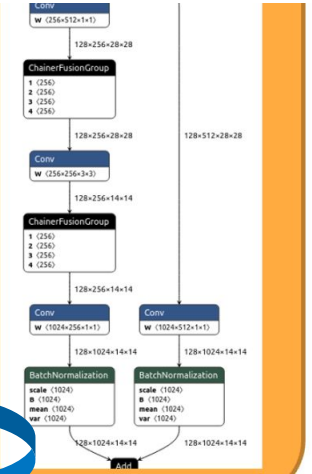
MN-Core MLSDK

- MLSDK(Machine Learning Software Development Kit)
  - MN-CoreでPyTorchのコードを動作させるためのSDK
  - 以下のものから構成される
    - Compiler
    - Python Interface
    - Visualization tool

MN-Core MLSDK

PyTorch → PFVM

ONNX model



L3IR: DNN op level, SIMD parallelism strategy, ……

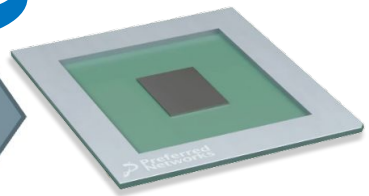
- MNGraph / MNNode / MNValue
- Global Layout Planner
- Re-computation Scheduler

L2IR/Generic Conv: ndarray op level, optimized DNN op impl, ……

- Generic Conv Impl
- MNTensor
- PEVector
- Generic Move Impl

L1IR: MN-core op level, memory allocation, scheduling, optimization, ……

- Layer Impl
- L1level instruction merger
- L1IR Scheduling Graph



Preferred Networks

MN-Core MLSDK

- Python Interface
  - GPUで使われているPyTorchのコードを、最小限の変更でMN-Coreにポーティングできるよう設計されている

context.registerは
tensor.to(“mncore”)
を意味する

context.compileは
torch.compileに相当

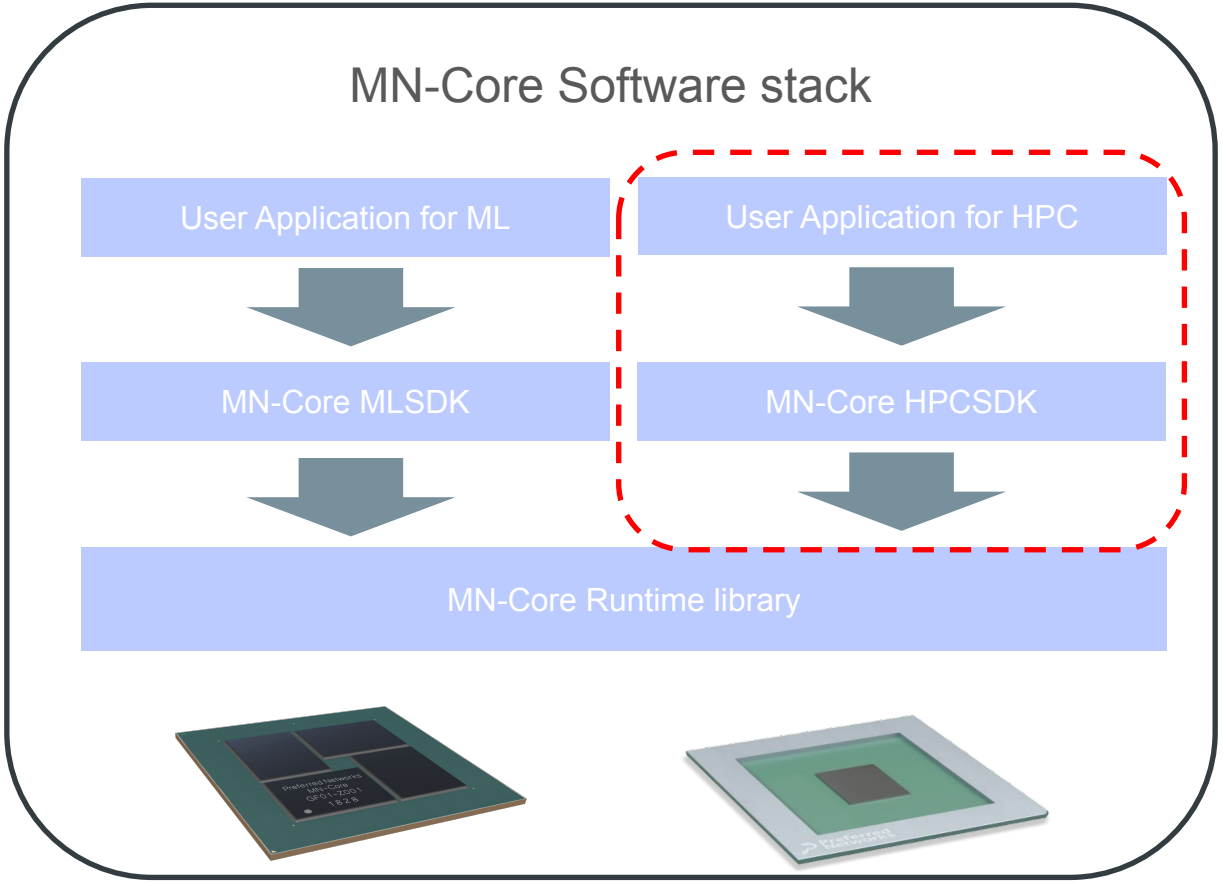
```
model.train()

def step(model, optimizer, data, target):
    optimizer.zero_grad()
    output = model(data)
    loss = F.nll_loss(output, target)
    loss.backward()
    optimizer.step()
    return loss

model = context.register(model)
optimizer = context.register(optimizer)
compiled_step = context.compile(step)

for (i, (d, t)) in enumerate(train_loader):
  loss = compiled_step(model, optimizer, d, t)
  if i % 1000 == 0:
      print(f”loss = {loss.cpu()}”)
```

MN-Core Software Stack



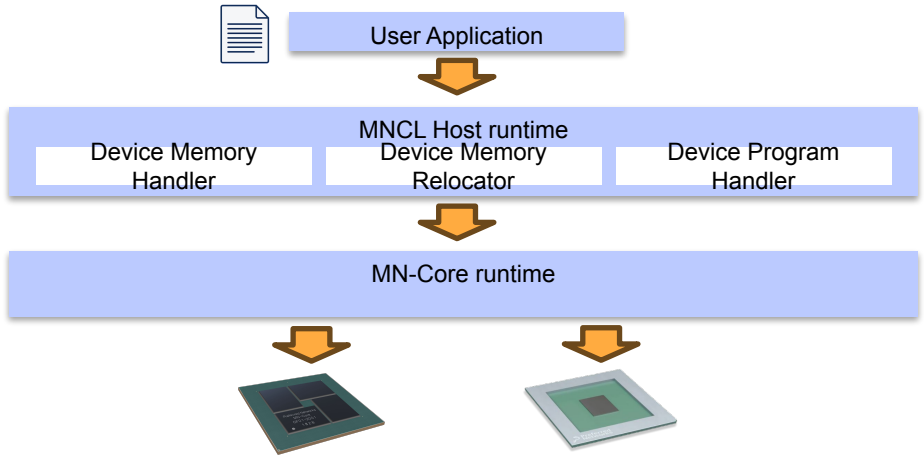
Preferred Networks

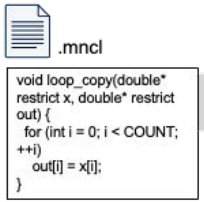
MN-Core HPCSDK

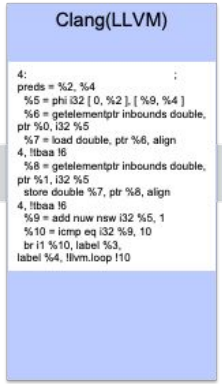
- HPCSDK(High-Performance Computing Software Development Kit)
  - MN-Coreで機械学習以外の汎用的な計算を行うためのSDK
  - 以下のものから構成される
    - MNCL
    - MNACC
    - Profiler
    - Debugger

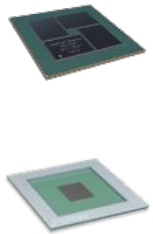
HPCSDK - MNCL

- **MNCLとはなにか？**
  - MN-Core 向けLow-levelなプログラミング環境
  - OpenCLの機能限定サブセットのように振る舞う





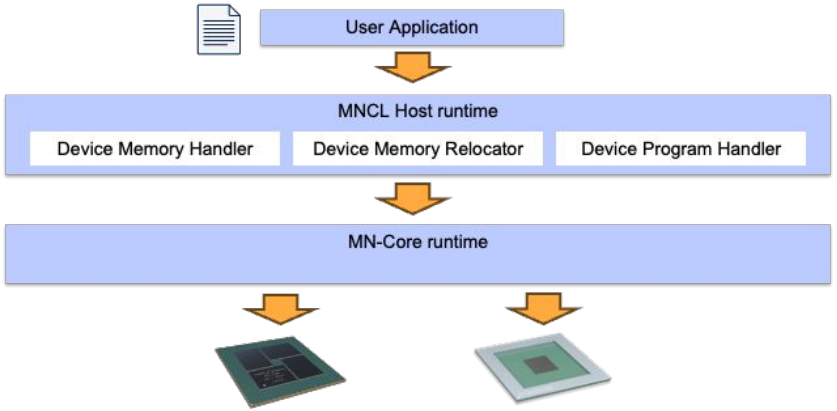




Preferred Networks

HPCSDK - MNCL

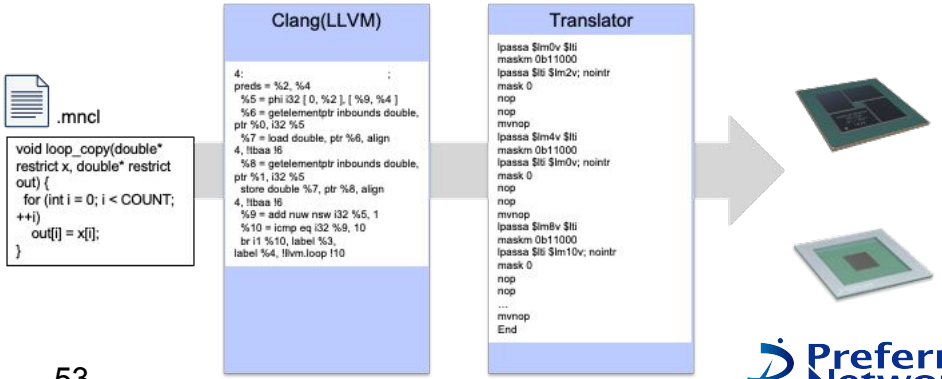
- MNCL Host runtime
  - OpenCLとほぼ同じAPIでデバイスをハンドルできる仕組み
- Key:
  - **OpenCLを書いたことがある方なら、MN-Coreを違和感なく扱うことができる**



Preferred Networks

HPCSDK - MNCL

- MNCL Device Translator
  - C/C++ のコードから、MN-Coreのアセンブリコードを生成する
    - LLVM-IRからコード生成を可能にする
  - レジスタ方向SIMD, サイクル方向SIMDを考慮したコード生成
  - Builtin functionによる階層間データ転送を記述可能

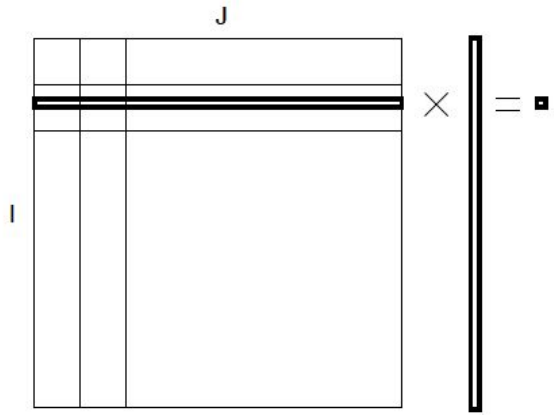


Preferred Networks

HPCSDK - MNACC

- MNACCとは？
  - Pragma で記述できる並列プログラミング環境
  - MN-Core 向けOpenACCのサブセット

```
#pragma goose parallel for loopcounter(i,j) result(c[i])
for(i=0;i<ni;i++) {
        c[i] = 0.0;
        for(j=0;j<nj;j++) {
                c[i]+=a[i][j]*b[j];
        }
}
```

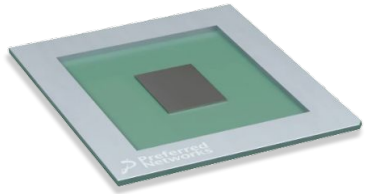


Preferred Networks

MN-Core benchmark

- MN-Core シリーズの性能は？

	MN-Core 2	A100
GCN(PFN internal use)	5.41TF(FP32)	3.17TF
ResNet50 training	77TF(FP16)	33.2TF(BF16)
ResNet50 Inference	154TF(FP16)	33.7TF(BF16)
HIMENO benchmark	9.03TF(FP32)	0.634TF
OpenFDTD	0.655TF(FP32)	0.488TF



- Himeno Benchmark
  - MN-Core 2 の Near-memory Computing Architectureが非常にハマった例

	GH200	MN-Core 2
メモリ帯域	4.8 TB/s	約 500 GB/s
理論性能	66.9 TFlops	12.28 TFlops
実行性能	**1.24 TFlops**	**9.025 TFlops**
実行効率	**1.8 %**	**73.4 %**

```
for(i=1 ; i<imax-1 ; ++i)
  for(j=1 ; j<jmax-1 ; ++j)
    for(k=1 ; k<kmax-1 ; ++k){
      s0 = a[i][j][k][0] * p[i+1][j  ][k  ]
         + a[i][j][k][1] * p[i  ][j+1][k  ]
         + a[i][j][k][2] * p[i  ][j  ][k+1]
         + b[i][j][k][0] * ( p[i+1][j+1][k  ] - p[i+1][j-1][k  ]
                           - p[i-1][j+1][k  ] + p[i-1][j-1][k  ] )
         + b[i][j][k][1] * ( p[i  ][j+1][k+1] - p[i  ][j-1][k+1]
                           - p[i  ][j+1][k-1] + p[i  ][j-1][k-1] )
         + b[i][j][k][2] * ( p[i+1][j  ][k+1] - p[i-1][j  ][k+1]
                           - p[i+1][j  ][k-1] + p[i-1][j  ][k-1] )
         + c[i][j][k][0] * p[i-1][j  ][k  ]
         + c[i][j][k][1] * p[i  ][j-1][k  ]
         + c[i][j][k][2] * p[i  ][j  ][k-1]
         + wrk1[i][j][k];

      ss = ( s0 * a[i][j][k][3] - p[i][j][k] ) * bnd[i][j][k];

      gosa = gosa + ss*ss;

      wrk2[i][j][k] = p[i][j][k] + omega * ss;
    }
```

- Himeno Benchmark
  - MN-Core 2 の Near-memory Computing Architectureが非常にハマった例

	GH200	MN-Core 2
メモリ帯域	4.8 TB/s	約 500 GB/s
理		
演		
演		

```
for(i=1 ; i<imax-1 ; ++i)
  for(j=1 ; j<jmax-1 ; ++j)
    for(k=1 ; k<kmax-1 ; ++k){
      s0 = a[i][j][k][0] * p[i+1][j  ][k  ]
         + a[i][j][k][1] * p[i  ][j+1][k  ]
         + a[i][j][k][2] * p[i  ][j  ][k+1]
         + b[i][j][k][0] * ( p[i+1][j+1][k  ] - p[i+1][j-1][k  ]
                           - p[i-1][j+1][k  ] + p[i-1][j-1][k  ] )
         + b[i][j][k][1] * ( p[i  ][j+1][k+1] - p[i  ][j-1][k+1]
                                                          -1][k-1] )
                                                           ][k+1]
                                                           ][k-1] )
                                                                  j][k];
      wrk2[i][j][k] = p[i][j][k] + omega * ss;
    }
```

MN-Core に向いていそうなアプリケーションを
他にもたくさん発掘していきたい

MN-Core の外部への展開

- MN-Core エコシステム構築のための試み

MN-Core 勉強会 #1

MN-Core Challange

Preferred Networks

今後の講習会の予定

PFNとFOCUS様は、今後もMN-Core 2を活用した講習会を共同で開催していく予定です

- 第二回 MNACC入門編
  - 2025/1/31(予定)
- 第三回 DeepLearning on MN-Core(仮題)
  - 2025/3月中(予定)

MN-Coreでこんな話を聞きたい、こんなアプリケーションが動くかを知りたいという要望があれば、お気軽に　PFNにまでご連絡いただけますと幸いです！

まとめ

- MN-Core シリーズのアーキテクチャと、開発環境についてご説明しました。
- FOCUS様にてこれより皆様にご活用いただけることを大変うれしく思います。
- ぜひ色々使い倒していただけることを願ってやみません！

謝辞

本発表でご紹介したMN-Core 2はNEDO（国立研究開発法人新エネルギー・産業技術総合開発機構）の委託業務（ JPNP16007）の結果得られたものです．また，本発表は，文部科学省「次世代計算基盤に係る調査研究」事業の助成を受けたものです．

Preferred Networks

Preferred Networks

Making the real world computable