MN-Core 2 Devkit
MN-Server 2

インストール・運用マニュアル

2026/04/23版

Preferred Networks

セットアップ

動作環境

MN-Core 2 Devkit / MN-Server 2では、以下のOSを必要動作環境としています。

- Ubuntu 24.04 LTS
- Ubuntu 22.04 LTS

また、運用時には必須ではありませんが、インストール時には、パッケージインストールのためにインターネットへの接続が必要です。

セットアップ手順

OSインストール

MN-Core 2 DevkitもしくはMN-Server 2に上記推奨のOSをインストールします。インストール用イメージを https://jp.ubuntu.com/download から入手し、USBメディアに焼いてインストール用USBメディアを作成してください。インストールに必要なUSBメディアは各自ご準備ください。

インストール用USBメディアをシステムに接続し、起動してインストールを進めます。起動するガイダンスに従い、標準の構成でインストールしてください。

OSのインストールのあとは、必要に応じてユーザーの追加とsudoersの設定を行ってください。以降の手順は、管理者のユーザーアカウントで作業し、権限が必要な場合は `sudo` コマンドを使う形式で示します。

必要パッケージのインストール

MN-Core 2開発環境に必要なパッケージをインストールします。ネットワーク接続が必要です。以下のコマンドを実行し、まずシステムのパッケージを最新に更新し、次に必要なパッケージをインストールします。

```
None

$ sudo apt update
$ sudo apt upgrade
$ sudo apt install ca-certificates git dkms pciutils ipmitool docker.io
docker-buildx
```

必要に応じて、開発環境の利便性のために、docker groupを作成してdockerコマンドをsudoなしで利用できるように設定することができます。詳細は https://docs.docker.com/engine/install/linux-postinstall/ を参照してください。設定の際は、信頼できるユーザーにのみdockerコマンドの利用を許可するよう注意してください。

pfnet/mncoreのクローン

MN-Core 2 Devkit/MN-Server 2に必要なユーティリティは、GitHubリポジトリ (https://github.com/pfnet/mncore) で配布されます。このリポジトリをホームディレクトリにクローンします。

```
None

$ git clone https://github.com/pfnet/mncore.git
```

以降の手順は、このディレクトリを `$HOME/mncore` で示します。

APTリポジトリの登録

pfnet/mncoreをクローンしたら、次にMN-Coer DriverおよびSDKを配布するAPTリポジトリをシステムに登録します。

```
None
$ cd $HOME/mncore
$ sudo bash apt/add_mncore_packages.sh
```

MN-Core 2ドライバおよびユーティリティのインストール

システムにAPTリポジトリを登録したら、次にMN-Core 2に必要なドライバおよびユーティリティをインストールします。

```
None
$ sudo apt update
$ sudo apt install gpfn3-dkms libgpfn3-dev gpfn3-smi
```

APTによるインストール完了後、エラーが出ていないことを確認したら再起動を行います。再起動後に以下のコマンドを実行し、MN-Core 2デバイス名が表示されれば、ドライバのインストールは完了です。

```
None
$ gpfn3-smi list
```

Devkitでは以下のように表示されます。

```
None
$ gpfn3-smi list
0: mnc2p1s0
```

MN-Serverでは以下のように表示されます。

```
None
$ gpfn3-smi list
0: mnc2p27s0
1: mnc2p28s0
2: mnc2p42s0
3: mnc2p43s0
4: mnc2p171s0
5: mnc2p172s0
6: mnc2p187s0
7: mnc2p188s0
```

MN-Core 2を使用する前の設定

MN-Core 2のスタートアップ

MN-Core 2は電源を投入するたびにクロック等の設定が揮発します。通常はkernel module (gpfn3-dkms) のロード時に設定が行われますが、以下のコマンドで手動で設定することもできます。`<devname>` はgpfn3-smi listコマンドが表示するデバイス名です。

```
None
$ gpfn3-smi config <devname> clock --core 750 --gddr6 15000
$ gpfn3-smi config <devname> mab
```

MN-Core 2の動作確認

MN-Core 2の動作確認は、以下のコマンドを実行します。

```
None
$ gpfn3-smi reset <devname>
$ gpfn3-smi mtest --forward --dmode h --l1 64 --l2 2048 --l3 64 --dram 3968
<devname>
```

gpfn3-smi mtestコマンドはMN-Core 2で1分ほどの計算を実行します。MN-Core 2が正常に動作した場合は、`All OK` のメッセージが表示され、終了コード0が返却されます。

MN-Core 2をリセットする

以下のコマンドを実行すると、実行中のDMAを停止し、メモリをクリアできます。
`<devname>` はgpfn3-smi listコマンドが表示するデバイス名です。

```
None
$ gpfn3-smi reset <device>
```

デバイスが正常にリセットできた場合はexit code 0 が返ります。

MN-Core SDKを利用する

この章では、MN-Core 2 Devkit / MN-Server 2にMN-Core SDKの開発環境を作成するための手順を説明します。コマンドは MN-Core SDK v0.4を利用する場合を示します。異

なるバージョンのMN-Core SDKを利用する場合は、各コマンドに含まれる`0.4`を利用するバージョンに置き換えて実行してください。

サポートされるコンテナエンジン

MN-Core SDKはコンテナ内での使用が推奨されます。MN-Core SDKをインストールした環境を作成し起動するためのコンテナエンジンとして、Dockerを使用します。Dockerのインストール手順は、セットアップ手順 ― 必要パッケージのインストールを参照してください。

Dockerの代わりに、root権限を要求しないコンテナエンジンであるPodmanを利用することもできます。Podmanの詳細は https://podman.io/ を参照してください。

いずれのコンテナエンジンを使う場合も、Ubuntu公式の最新版パッケージ (`docker.io` もしくは `podman`) でインストールしたものをご利用ください。

MN-Core SDKコンテナイメージの作成

以下のコマンドを実行し、MN-Core SDKをインストールしたコンテナイメージを作成します。`$HOME/mncore` はセットアップ手順 ― pfnet/mncoreのクローンでクローンしたディレクトリです。

```
None

$ cd $HOME/mncore/sdk/0.4
$ docker build -t mncore-sdk-minimal:0.4 -f mncore-sdk-minimal.Dockerfile .
$ docker build -t mncore-sdk-full:0.4 -f mncore-sdk-full.Dockerfile --build-arg
minimal_image_ref=mncore-sdk-minimal:0.4 .
```

ビルドしたコンテナイメージは以下のコマンドで確認します。

```
None
$ docker image ls
```

出力に `mncore-sdk-full:0.4` が含まれていれば、MN-Core SDKコンテナイメージは正常にビルドされています。

MN-Core SDKコンテナの起動

MN-Core SDKを使う環境を起動するには、以下のコマンドを実行してpodmanコンテナインスタンスを起動します。<devname> はgpfn3-smi listコマンドが表示するMN-Core 2デバイス名です。ここで指定したデバイスがコンテナにマウントされ、コンテナ内から利用可能になります。

```
None
$ cd $HOME/mncore
$ bash sdk/0.4/create_dev_ctr.sh /dev/<devname>
Starting container
511a1e1782d200e2ffdd9e7f7d92ba428717c2ea86fbbf03669e2cca284c8360
Setting up devices
  /dev/<devname>
Setup succeeded. Launch a shell in the container by "podman exec --privileged
-it 511a1e1782d2 bash", or shutdown it by "podman stop 511a1e1782d2".
```

コンテナが正常に起動すると、標準出力にコンテナIDが表示されます。以下のコマンドでコンテナインスタンスにシェルを起動できます。コンテナ内のシェルからもMN-Core 2の動作確認を実行することができます。

```
None
$ docker exec --privileged -it <container_id> bash
```

コンテナを使い終わったときは、以下のコマンドでコンテナインスタンスを停止します。

```
None
$ docker stop <container_id>
```

MN-Core SDKの動作確認

`docker exec`で起動したコンテナ内のシェルで以下のコマンドを使用すると、MN-Core SDKのサンプルプログラムを実行し、MN-Core SDKが正しくインストールされているかを確認できます。

```
None
bash /opt/pfn/pfcomp/codegen/MLSDK/examples/run_mnist_on_torch.sh
```

実行すると、MN-Coreを使ってmnistの学習が走ります。正常に実行されたときの典型的な出力は以下のようになります。

```
None
# bash /opt/pfn/pfcomp/codegen/MLSDK/examples/run_mnist_on_torch.sh
+++ dirname /opt/pfn/pfcomp/codegen/MLSDK/examples/run_mnist_on_torch.sh
++ realpath /opt/pfn/pfcomp/codegen/MLSDK/examples
+ MLSDK_DIR=/opt/pfn/pfcomp/codegen/MLSDK/examples
+ pushd /opt/pfn/pfcomp/codegen/MLSDK/examples
/opt/pfn/pfcomp/codegen/MLSDK/examples /
... (途中省略) ...
epoch 9, loss 0.1358569860458374
+ ./exec_with_env.sh python3 mnist_infer.py
/tmp/mlsdk_mnist_train/checkpoint.pt
+++ dirname ./exec_with_env.sh
++ realpath .
+ CURRENT_DIR=/opt/pfn/pfcomp/codegen/MLSDK/examples
++ realpath /opt/pfn/pfcomp/codegen/MLSDK/examples/../../
+ CODEGEN_DIR=/opt/pfn/pfcomp/codegen
+ BUILD_DIR=/opt/pfn/pfcomp/codegen/build
+ source /opt/pfn/pfcomp/codegen/build/codegen_pythonpath.sh
++ export
PYTHONPATH=/opt/pfn/pfcomp/codegen/build/python_interface/:/opt/pfn/pfcomp/code
gen/python_trainer/mncore:/opt/pfn/pfcomp/codegen/python_trainer/codegen:/opt/p
fn/pfcomp/codegen/../fx2onnx:/opt/pfn/pfcomp/codegen/MLSDK/src:/opt/pfn/pfcomp/
codegen/../pfvm
++
PYTHONPATH=/opt/pfn/pfcomp/codegen/build/python_interface/:/opt/pfn/pfcomp/code
gen/python_trainer/mncore:/opt/pfn/pfcomp/codegen/python_trainer/codegen:/opt/p
fn/pfcomp/codegen/../fx2onnx:/opt/pfn/pfcomp/codegen/MLSDK/src:/opt/pfn/pfcomp/
codegen/../pfvm
+ exec python3 mnist_infer.py /tmp/mlsdk_mnist_train/checkpoint.pt
Correct: 9530 / 10000. Accuracy: 0.953000009059906
```

MN-Core SDKの活用

MN-Core SDKを用いてアプリケーションポーティングを行う場合には、
github.com/pfnet/mncore に掲載されている各種リソース/ドキュメントをご活用ください。

運用時の注意点

Devkit

電源投入時の注意点

MN-Core 2 Devkitは、一度電源を切った状態からの起動時にLinux kernelがMN-Core 2デバイスを認識しない可能性があります。その場合は、shutdownコマンドでシステムの再起動を行ってください。

Linux kernelがMN-Core 2デバイスを認識しているかどうかは、以下のコマンドで確認できます。

```
None
$ lspci | grep Preferred
```

デバイスが正常に認識されている場合は、以下の出力が得られます。認識されていない場合は、空の出力が得られます。

```
None
$ lspci | grep Preferred
01:00.0 Processing accelerators: Preferred Networks, Inc. MN-Core 2
```

MN-Server 2

ファン回転数の設定

MN-Server 2では、2024/10現在、負荷状況に応じてファンの回転数を制御する機能が実装されていません。MN-Core 2の熱暴走を防止するために、ファンの回転数を常に70%以上に設定することを推奨します。以下のコマンドを実行すると、ファンの回転数を100%に設定できます。

```
None
$ sudo ipmitool raw 0x30 0x45 0x01 0x01
```

トラブルシューティング

MN-Core 2デバイスが見えないとき

プログラムの実行が、デバイスの不在により失敗するときの対応方法を順に説明します。

デバイスファイルが見えるかを確認する

以下のコマンドを実行し、デバイスファイルがあるか確認します。デバイスファイルは、`/dev/mnc2p32s0` といった形式です。

```
None
$ ls /dev/mnc2p*
```

デバイスファイルが見えない場合、以下の可能性があります。

- MN-Core 2 kernel module (gpfn3-dkms) が正しくインストール / ロードされていない
- MN-Core 2デバイスが故障している

MN-Core 2 kernel moduleがインストール / ロードされているか確認する

MN-Core 2 kernel moduleのインストール状況を確認します。以下のコマンドを実行します。

```
None
$ sudo dpkg-query -l gpfn3-dkms
```

行先頭に表示されるステータスが `ii` であればMN-Core 2 kernel moduleが正しくインストールされています。ステータスがそれ以外の場合、kernel moduleが正しくインストールされていません。セットアップ手順 — MN-Core DriverおよびユーティリティのインストールをMN-Core 2 kernel moduleを参照し、kernel moduleを再度インストールしてください。インストール後は再起動が必要です。

MN-Core 2 kernel moduleが正しくインストールされている場合は、ロードされているかを確認するために以下を実行します。

```
None
$ sudo lsmod | grep gpfn3
```

正しくロードされている場合、`gpfn3`が表示されます。表示されない場合はインストール・セットアップの項を参照し、MN-Core 2 kernel moduleを再度インストールしてください。インストール後は再起動が必要です。

デバイスの**PCIe** エンドポイントが動作しているか確認する

MN-Core 2 kernel moduleがロードされているがデバイスファイルが見えない場合、まずMN-Core 2のPCIeエンドポイントが動作しているか確認します。 以下のコマンドを実行します。`0ccd` はPFNのvendor idです。

```
None
$ sudo lspci -d 0ccd:
```

Linux kernelが認識しているPCIeエンドポイントが表示されます。表示された数がシステムに搭載されているボード数と一致するか確認してください。

- 一致していない場合
  - まずマシンの再起動を試みてください
    - システム起動時に、デバイスがreadyになる前にOSがPCIeエンドポイントをスキャンするせいでPCIeエンドポイントが認識されないままとなることがあります。これはほとんどの場合、システムを再起動することで解決します
  - 数回再起動してもPCIeエンドポイントが認識されない場合は、MN-Coreボード上のチップが故障しています。ボード交換が必要なので、lspciの結果を添付してPFN担当者に問い合わせてください
- 一致している(が、デバイスファイルが足りない)場合 kernel module のlogを確認する、に進みます

MN-Core 2 kernel moduleのログを確認する

kernelが認識しているPCIeエンドポイント数がシステムに搭載されたボード数に一致し、いずれかのデバイスファイルが見えていない場合は、Linux kernel logを調査してMN-Core 2 kernel moduleが正しく動いているかを確認します。以下のコマンドを実行します。

```
None
$ sudo journalctl | grep gpfn3
```

grep結果の末尾を見て、各デバイスについて最後のノード起動時に正常に認識したかを確認します。

- Linux kernelがPCIeエンドポイントを認識し、MN-Core 2 kernel module (gpfn3)が正常に初期化を終えたデバイスについては、`kernel: gpfn3 0000:01:00.0: mnc2p32s0 loaded.`のログが残ります
- Linux kernelがPCIeエンドポイントを認識し、MN-Core 2 kernel module (gpfn3)が初期化に失敗したデバイスについては、何らかのエラーメッセージが残ります
    - MN-Core 2 kernel moduleのバグの可能性があるので、grep結果を添付してPFN担当者にお問い合わせください
- Linux kernelがPCIeエンドポイントを認識しなかったデバイスについてはログに表示されません
    - デバイスのPCIeエンドポイントが動作しているか確認する に戻ります

MN-Core 2が動作しないとき・挙動が疑わしいとき

MN-Core 2のプログラムが正しく動作しないときは、まずMN-Core 2が正常な設定で動作していないことを疑います。MN-Core 2を使用する前の設定に従い、デバイスのセットアップとリセットを試みてください。デバイスのセットアップとリセットに成功したら、再度プログラムの実行を試みてください。

お問い合わせ

この製品の使い方や修理に関するお問い合わせは、下記サポート連絡用メールアドレスにメールにてご連絡ください。

mn-core-support@preferred.jp

【お願い】
ご連絡いただく場合には、おわかりになる範囲で結構ですので、次の内容をご記載いただきたくお願いします。

- 問題が発生した時の状況

- 表示されたメッセージ
- 症状の発生頻度
- OSの品名、バージョン
- その他、発生に関わると思われる情報

※本製品は弊社で開発した MN-Core 2 チップと一般的なPC部品を組み合わせて作っています。一般的なPC部品の障害であった場合には、弊社でも解決できないトラブルの可能性もありますので対応に時間をいただく場合がございます。あらかじめご了承ください。

Revision History

- 2026/04/23
  - 手順の不備を修正・微細な表現を修正
- 2026/03/31
  - セットアップ手順をpfnet/mncoreリポジトリとAPTリポジトリを使うものに改訂
- 2026/02/25
  - バージョンの記載などを修正
- 2025/03/06
  - MN-Core SDKの運用方法をuDockerからpodmanに変更
- 2025/01/30
  - MN-Core SDKのイメージ名をminimalからfullへ変更
- 2025/01/28
  - MN-Core SDK導入について記載
- 2024/10/24
  - 初版