AI on IBM Power:
Learn How IBM Power Can Solve your AI Challenges

Suyog Jadhav
Rocket Software Senior Manager
sujadh23@in.ibm.com

Rajalakshmi Srinivasaraghavan
Linux on Power Toolchain
rajalakshmi.srinivasaraghavan@ibm.com

Alexander Lang
Architect
alexlang@de.ibm.com

Si Win
Data and AI on Power, PM
stwin@us.ibm.com

Agenda

- Where to Start for AI on Power - RocketCE – Suyog Jadhav
  - What is RocketCE
  - How to Obtain RocketCE
  - How to Stay Informed
  - How to Use RocketCE
- How Power is beneficial for AI - AI Acceleration using MMA in P10 - Rajalakshmi Srinivasaraghavan
  - Overview of MMA
  - MMA enabled AI Libraries
  - Benefits of MMA
- How to Participate and Influence the Open Source Process - OpenCE Update - Alexander Lang
  - Build Conda Packages
  - Optimized Build Recipes for P10

RocketCE : OpenCE For Power

Suyog Jadhav

Senior Manager, IBM Channel Power Products

IBM Champion

Agenda

- What is RocketCE
- How To Obtain RocketCE
- How To Stay Informed
- How To Use RocketCE

Prerequisites

- Linux
- Python 3.x
- Conda Package Manager
    - https://docs.conda.io/

What Is RocketCE (Rocket Cognitive Environment)

- RocketCE is Set of AI/ML Conda Packages Optimized For Power Platform
  - Tensorflow, Pytorch, OnnxRuntime
- Solves problem of setting up AI/ML Environments
  - Create environment from scratch
  - Clone previous environment
- Conda can setup environment taking care of all dependencies

How to obtain RocketCE

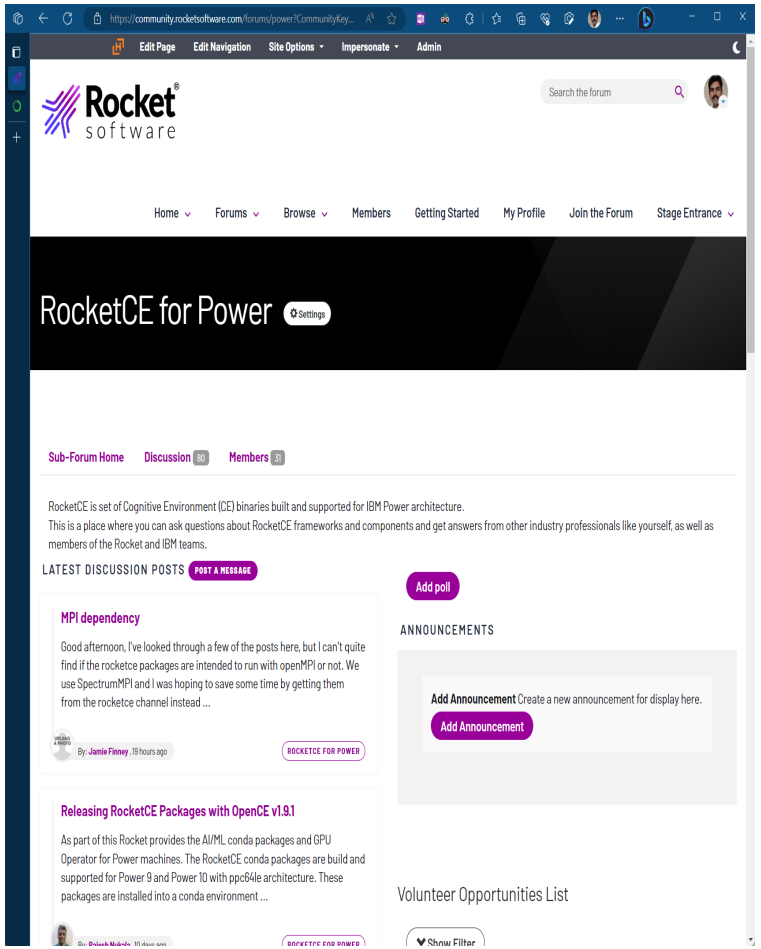
- Conda Packages
  - https://anaconda.org/rocketce
- PIP Location
  - https://pypi.org/project/onnxruntime-powerpc64le
- Containers
  - https://quay.io/organization/rockece

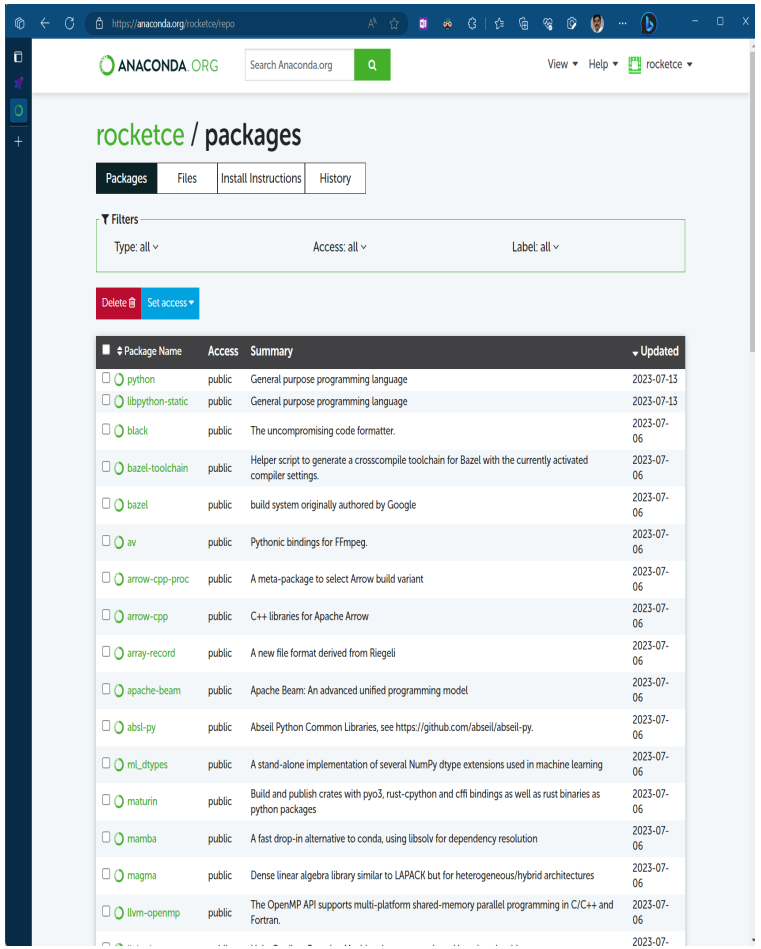
How to Stay Informed

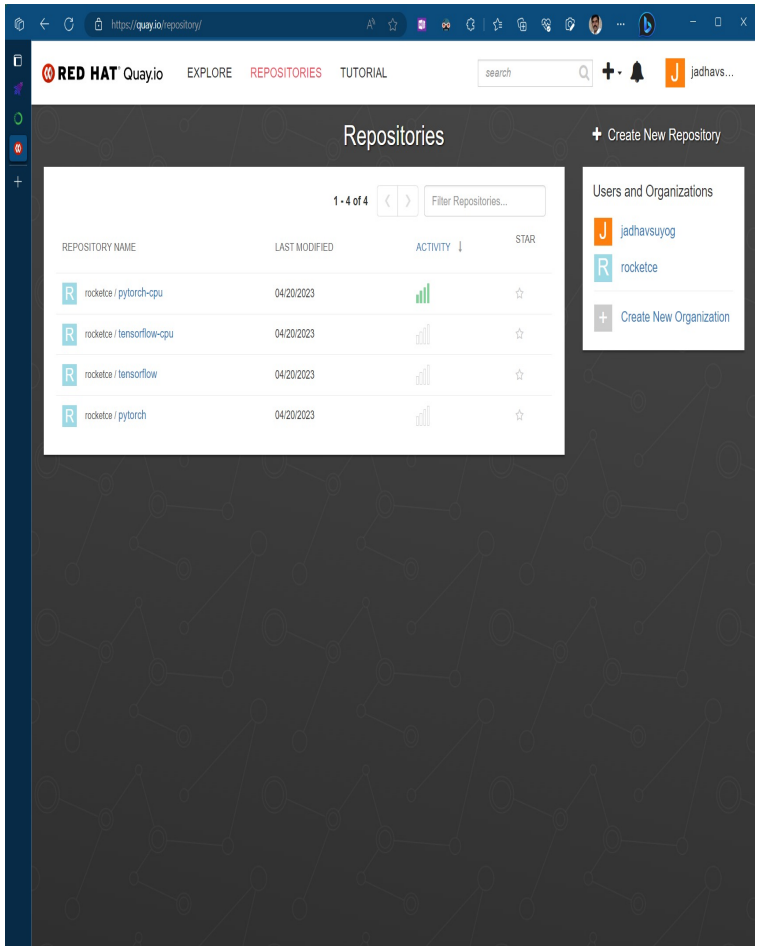
- Join Rocket Software Forum
  - https://community.rocketsoftware.com/home

- Subscribe to RocketCE community
  - https://community.rocketsoftware.com/forums/power?CommunityKey=c7ece6e8-5a29-4a17-a2bc-68b65f89d29f

Important sites







How to use RocketCE

- Install Conda package manager
- Create a new conda environment
  - conda create -n myEnv --python=3.10
- Activate the newly created environment
  - conda activate myEnv
  - conda list
- Install required package
  - conda install -c rocketce tensorflow
  - conda install -c rocketce pytorch
- Write Your Awesome tool/script !

Agenda

- Where to Start for AI on Power - RocketCE – Suyog Jadhav
  - What is RocketCE
  - How to Obtain RocketCE
  - How to Stay Informed
  - How to Use RocketCE
- How Power is beneficial for AI - AI Acceleration using MMA in P10 - Rajalakshmi Srinivasaraghavan
  - Overview of MMA
  - MMA enabled AI Libraries
  - Benefits of MMA
- How to Participate and Influence the Open Source Process - OpenCE Update - Alexander Lang
  - Build Conda Packages
  - Optimized Build Recipes for P10

AI Acceleration using MMA in P10

Rajalakshmi Srinivasaraghavan

Matrix Multiply Assist in POWER ISA

- MMA architecture support is introduced in POWER ISA V3.1.
- MMA architecture introduces new set of instructions to support dense matrix math operations along with required changes for register handling and management.
- Most operations in training/inferencing in a neural network require some form of matrix multiplication.
- These Matrix-Multiply Assist instructions lead to very efficient implementations for key algorithms in technical computing, machine learning, deep learning and business analytics, it is a natural match for implementing **dense numerical linear algebra computations. Example:** GEMM - General Matrix to Matrix Multiplication – multiply two matrices

MMA support in compilers

MMA support has been enabled in GCC/Clang using built-ins

Built-in type	Description
__vector_quad	Accumulator data type.
__builtin_mma_xxsetaccz()	Reset accumulators.
__builtin_mma_build_acc() __builtin_mma_disassemble_acc()	Merge/Disassemble accumulators
__builtin_mma_xvf*ger*()	All precision Matrix multiply or multiply accumulate or negate-accumulate.
__builtin_mma_pxvf*ger*()	Prefixed/Masked matrix multiply operations.
__builtin_vsx_build_pair() __builtin_vsx_disassemble_pair()	Pair/Unpair register set (used for dgemm)

Full list of supported built-ins is available in the following link

https://gcc.gnu.org/onlinedocs/gcc/PowerPC-Matrix-Multiply-Assist-Built-in-Functions.html

Programming using builtins

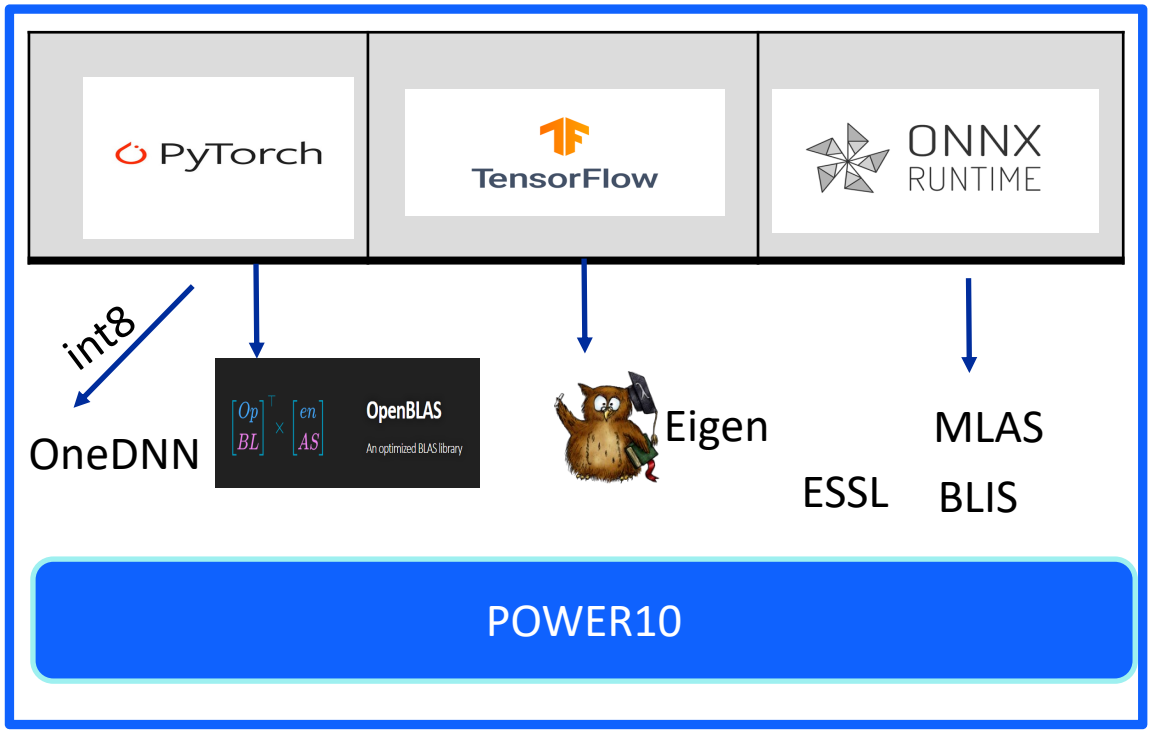
```
void
foo (vec_t *A, vec_t *B, vec_t *C)
{
    __vector_quad acc0, acc1;
    vector unsigned char result[4];

    __builtin_mma_xxsetaccz (&acc0);
    __builtin_mma_xxsetaccz (&acc1);

    for (int i = 0; i < 8; i += 2)
      {
        __builtin_mma_xvf32gerpp (&acc0, A[i], B[i]);
        __builtin_mma_xvf32gerpp (&acc1, A[i+1], B[i+1]);
      }

    __builtin_mma_disassemble_acc (result, &acc0);
    C[0] = result[0];
    C[1] = result[1];
    C[2] = result[2];
    C[3] = result[3];
    __builtin_mma_disassemble_acc (result, &acc1);
    C[4] = result[0];
    C[5] = result[1];
    C[6] = result[2];
    C[7] = result[3];
}
```

POWER10 MMA support in frameworks



OpenBLAS

- MMA support has been enabled in latest OpenBLAS for POWER10.
- Support available for Float, Double, Complex, Real GEMM and TRMM kernels.
- Easy integration possible with Python-NumPy library, PyTorch and other frameworks which uses OpenBLAS for BLAS to exploit P10 MMA.
- bfloat16 – reduced size and highly adopted in ML/DL
- Support added in OpenBLAS and optimized for Power10
- Level 1(vector-vector) and Level 2 (Matrix-vector) functions optimized to make use of P10 vector pair instructions.
- Exploitation of current and future designs of MMA made easy
  - Converted handwritten assembly version used in previous versions for GEMM optimization to C built-ins

Eigen & ONNXRuntime

Eigen

- Design change to accommodate MMA - New packing introduced for POWER10.
- Level3 (matrix-matrix) for complex and real float/double and bfloat16 optimized for P10.

ONNXRuntime

- High performance runtime for ONNX models.
- Single precision float32 (SGEMM) , float64(DGEMM) and int8 (QGEMM) optimized for POWER10 using MMA.

POWER10 MMA Support in Libraries

Library	Version	Optimization
OpenBLAS **(Used in PyTorch, Numpy)** https://github.com/xianyi/OpenBLAS/	0.3.13 and above	MMA **Level 3** GEMM functions optimized: • Sgemm [float] • Dgemm [double] • Cgemm [complex float] • Zgemm [complex double] • Sbgemm (BFloat16) **Level 2** GEMV functions optimized for double type. • dgemv optimized to use power10 vector pair instructions. • Zgemv optimized using MMA **Level 1 vector-vector** functions optimized to use power10 vector pair instructions.
Eigen **(Used by Tensorflow)** https://gitlab.com/libeigen/eigen/	3.4	MMA **Level 3** GEMM (matrix-matrix) functions optimized for • Real float and double • complex float and double • bfloat16 MMA & VSX Level 2 GEMV (matrix-vector) functions optimized • Real float and double • complex float and double
ONNXRuntime https://github.com/microsoft/onnxruntime	1.9.0 and above	MMA **Level 3** GEMM functions optimized. • Sgemm • Dgemm in ORT 1.10 • Low precision: Int8 GEMM in 1.11
Numpy https://github.com/numpy/numpy/	1.23.0	• MMA for GEMM comes through OpenBLAS • Integer logical operation – and, or, not • Integer arithemetic operation – floor, fmod, divide, reminder • Integer comparison operation - greater, less than, equal
OneDNN https://github.com/oneapi-src/oneDNN/	2.7	• Low precision: Int8 GEMM with MMA (from 2.7 version) • MMA for bf16 , float and double comes from OpenBLAS
BLIS https://github.com/flame/blis	0.9.0	MMA **Level 3** GEMM functions optimized. –Sgemm & Dgemm Low precision GEMM functions introduced and optimized.[sandbox] • Bfloat16 / float16 • Int16 / int8 /int4

- Minimum compiler version gcc10.2 or clang12 needed to compile these libraries.
- Libraries (like eigen) can be directly built from source from community repository.
- Frameworks enabled with P10 MMA for python 3.8 are also available at https://anaconda.org/rocketce/repo
  - conda install -c rocketce pytorch-cpu
  - conda install -c rocketce tensorflow-cpu
  - conda install -c rocketce onnxruntime
  - conda install -c rocketce openblas

P10 aware AI frameworks

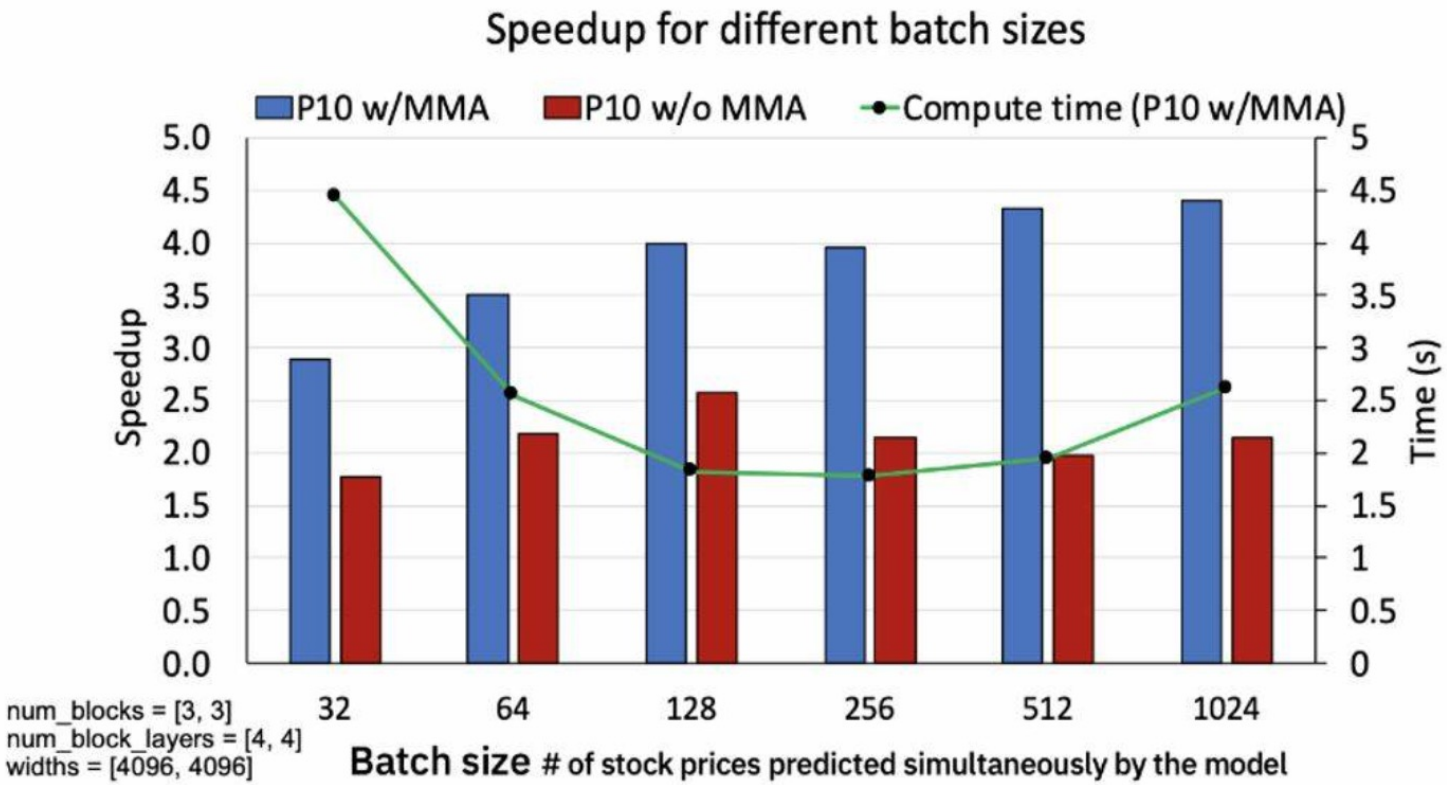
CPU only packages enabled with P10 MMA for Python 3.8 & above are available at:

https://anaconda.org/rocketce/repo

Install instructions are as follows.

- conda install -c rocketce pytorch-cpu
- conda install -c rocketce tensorflow-cpu
- conda install -c rocketce onnxruntime
- conda install -c rocketce openblas

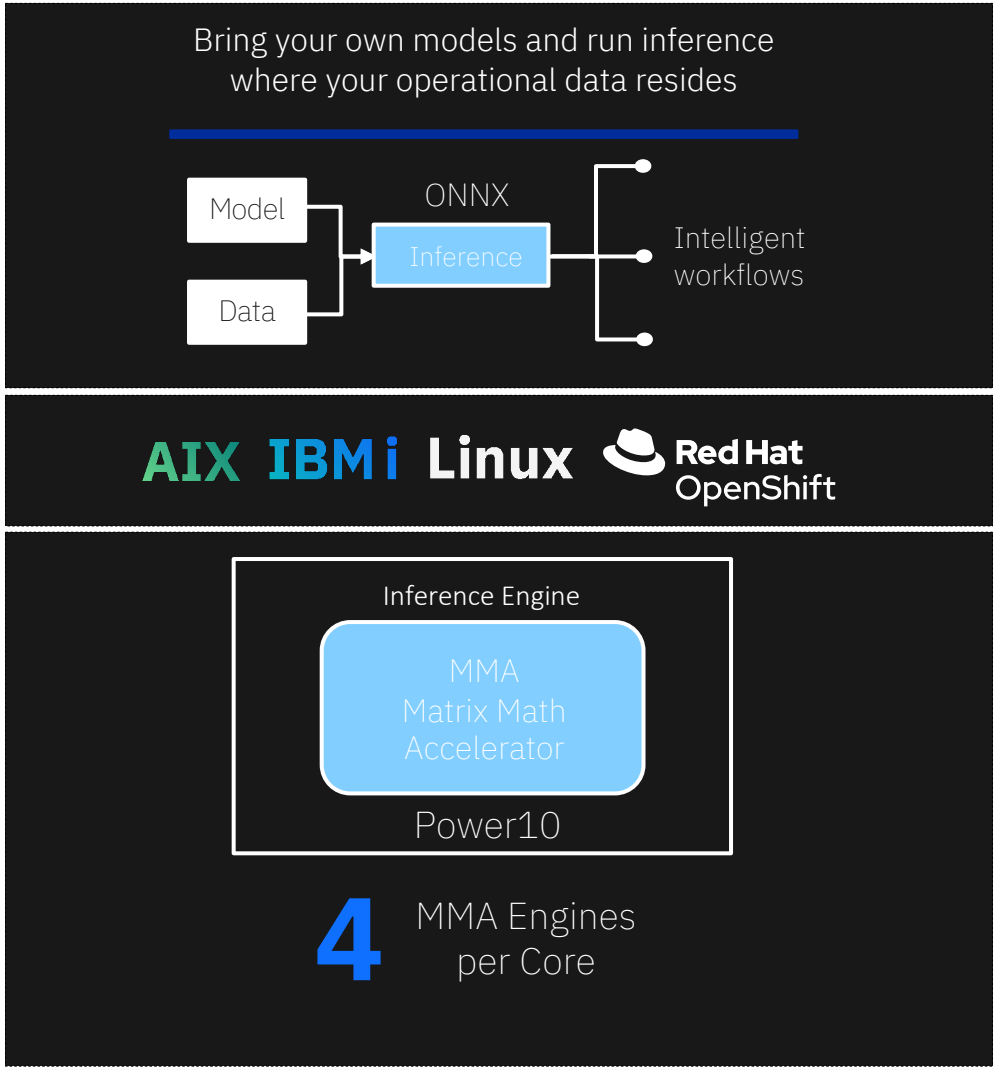
N-beats model results



N-BEATS* model from pytorch_forecasting. Yahoo dataset.
Model parameters: num_blocks, num_block_layers , widths . 4x with Batch size = 128
*N-BEATS: **N**eural **B**asis **E**xpansion **A**nalysis for interpretable **T**ime **S**eries forecasting, Oreshkin et al. https://arxiv.org/abs/1905.10437

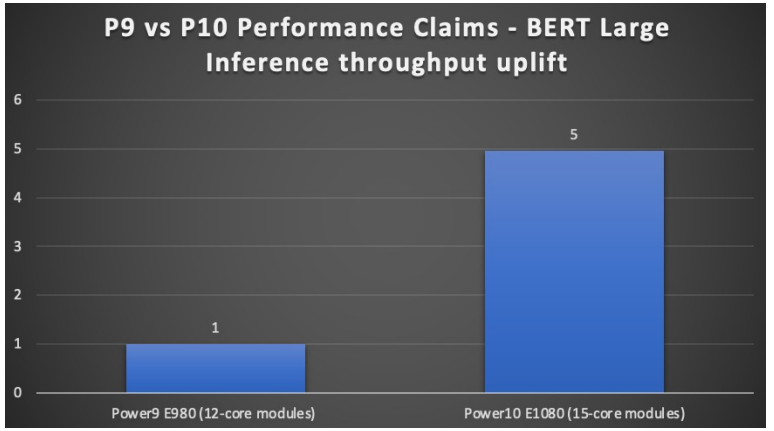
Reference : https://developer.ibm.com/tutorials/power10-business-inferencing-at-scale-with-mma/

In core AI inferencing and machine learning



5X

Faster AI inferencing per socket vs Power E980*



- Perform in-core AI inferencing and ML where data resides
- Provides alternative to using separate GPU systems
- Train AI models anywhere, deploy on Power without changes for AI with high RAS
- Support for popular libraries, AI frameworks and ONNX runtime

*5x improvement in per socket inferencing throughput for large size 32b floating point inferencing models from Power9 E980 (12-core modules) to Power10 E1080 (15-core modules). Based on IBM testing using Pytorch, OpenBLAS on the same BERT Large with SqUAD v1.1 data set

References

- https://gcc.gnu.org/onlinedocs/gcc/PowerPC-Matrix-Multiply-Assist-Built-in-Functions.html
- https://github.com/xianyi/OpenBLAS/tree/develop/kernel/power
- https://gitlab.com/libeigen/eigen/-/blob/master/Eigen/src/Core/arch/AltiVec/MatrixProductMMA.h
- https://www.redbooks.ibm.com/abstracts/redp5612.html?Open
- https://developer.ibm.com/blogs/run-ai-inferencing-on-power10-leveraging-mma/
- https://github.com/microsoft/onnxruntime

Agenda

- Where to Start for AI on Power - RocketCE – Suyog Jadhav
  - What is RocketCE
  - How to Obtain RocketCE
  - How to Stay Informed
  - How to Use RocketCE
- How Power is beneficial for AI - AI Acceleration using MMA in P10 - Rajalakshmi Srinivasaraghavan
  - Overview of MMA
  - MMA enabled AI Libraries
  - Benefits of MMA
- How to Participate and Influence the Open Source Process - OpenCE Update - Alexander Lang
  - Build Conda Packages
  - Optimized Build Recipes for P10

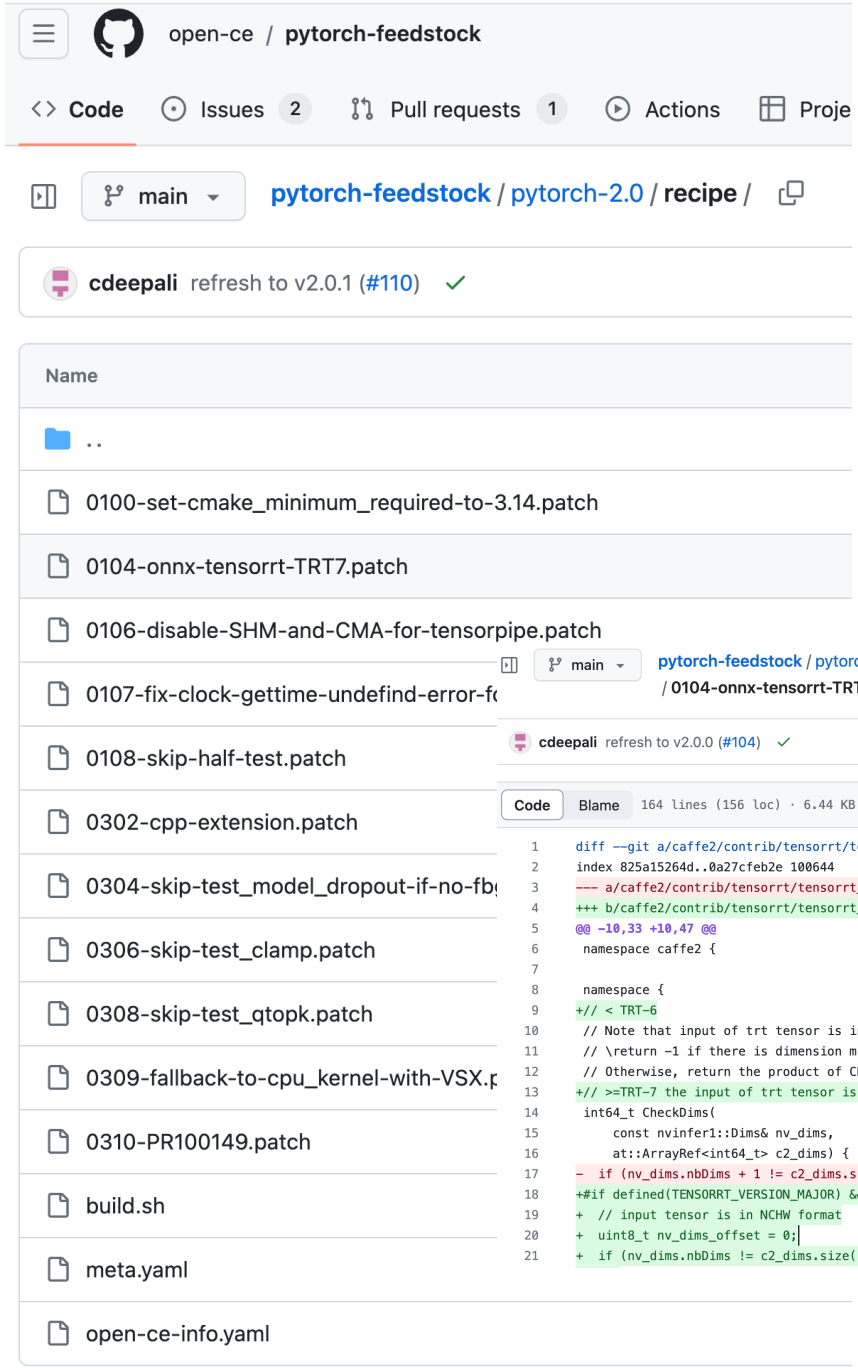
OpenCE

The cookbook for AI on Power

Conda build recipes

A **recipe** folder contains the information to build a conda package for a specific library

- meta.yaml contains dependencies on other libraries, at build time and run time
- build.sh contains the actual build command, including compiler settings
- .patch files contain updates to library code to fix issues on a particular target platform – or to add target-specific enhancements



https://docs.conda.io/projects/conda-build/en/latest/concepts/recipe.html

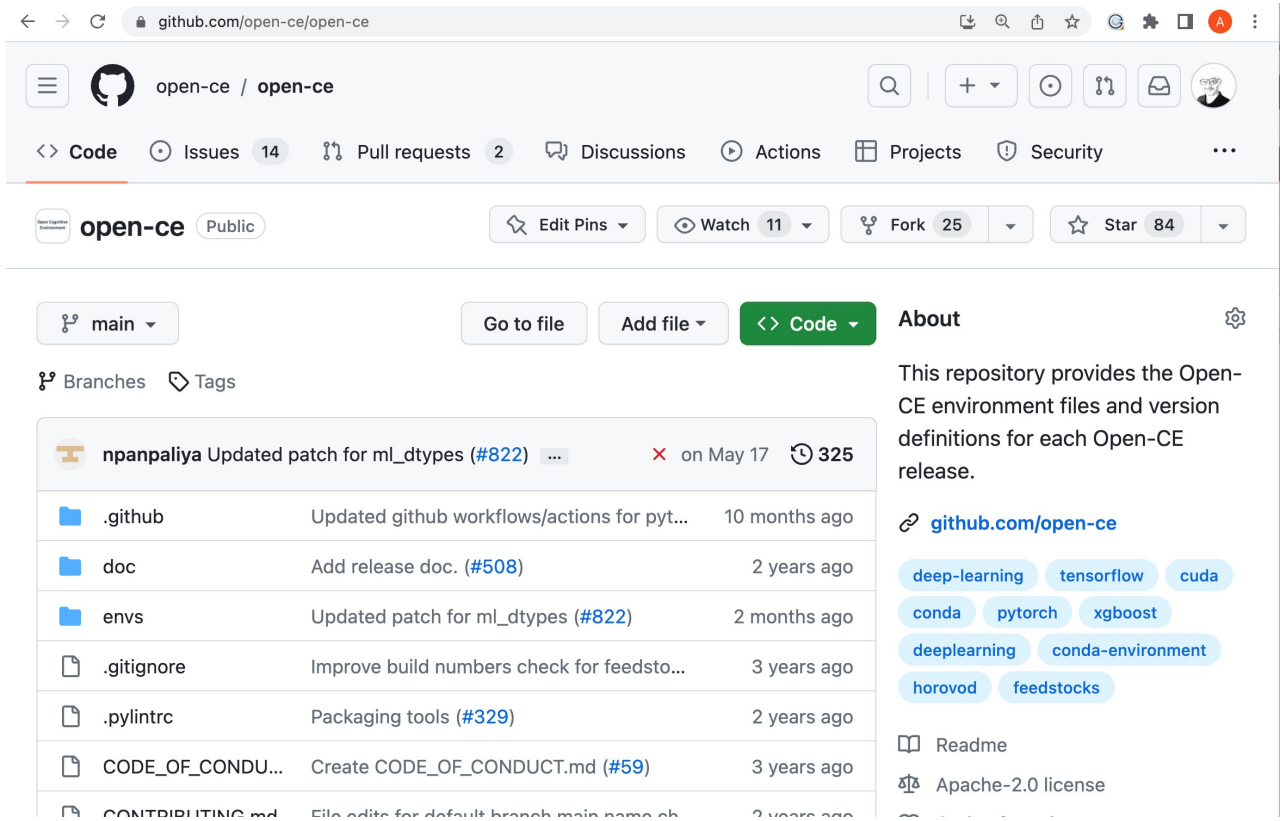
IBM OpenCE: optimized conda build recipes for Power https://github.com/open-ce/open-ce

Dedicated team that creates tools and recipes to build Python data science libraries for Power

- Ecosystem of build partners: **Rocket**, Oregon State University,...

Aligns *all library* dependencies across the `meta.yaml` files, so you can install PyTorch, Tensorflow, Ray,.. *into the same conda environment*

- Patches the existing open-source build recipes as needed



PyTorch, Tensorflow, Jax, deepspeed, ray, beam, mamba, prophet, xgboost,...

OpenCE: Advantages

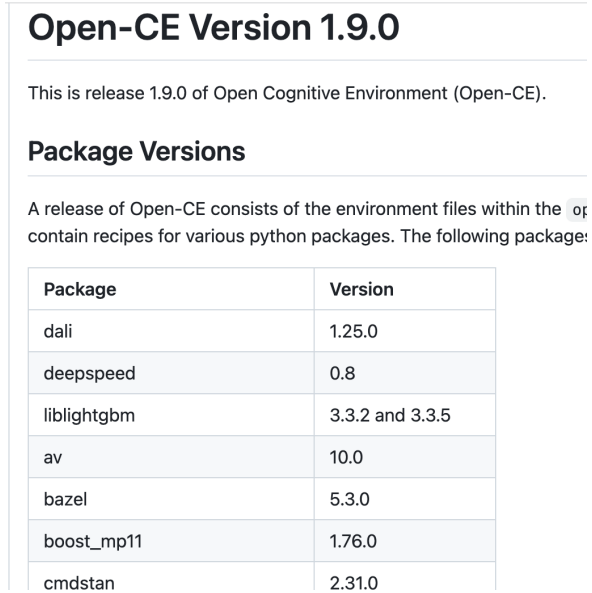
Optimized build recipes for Power10

Ongoing **Security** updates

- Pick up latest security fixes from the open-source community
- Team creates patches for TF, PyTorch, … if needed
- Team backports security fixes from newer releases of data science libraries

Quarterly major releases provide the latest major data science libraries

Regular, **non-breaking refreshes** provide minor-level library updates



Open-CE Version 1.9.0

This is release 1.9.0 of Open Cognitive Environment (Open-CE).

Package Versions

A release of Open-CE consists of the environment files within the o
contain recipes for various python packages. The following package

Package	Version
dali	1.25.0
deepspeed	0.8
liblightgbm	3.3.2 and 3.3.5
av	10.0
bazel	5.3.0
boost_mp11	1.76.0
cmdstan	2.31.0

https://github.com/open-ce/open-ce/releases

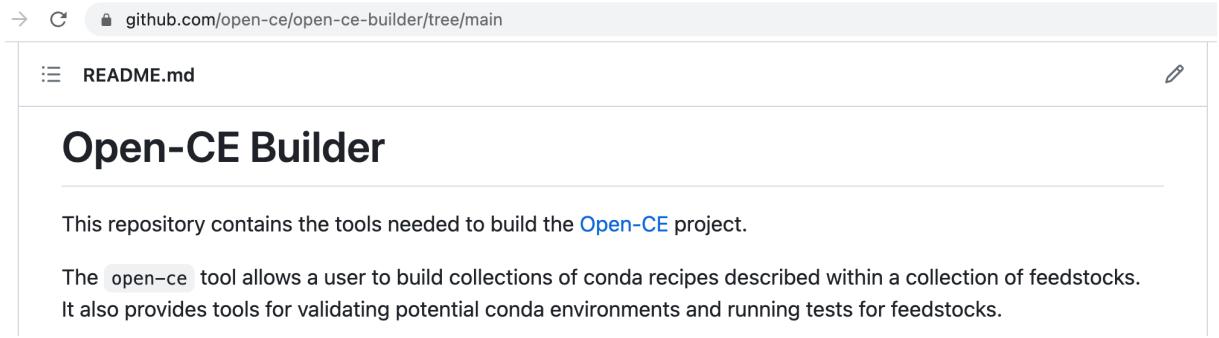
Build it yourself

https://github.com/open-ce/open-ce-builder

We think it's ***easiest*** to get the OpenCE conda packages from RocketCE – they're free and up-to-date, with no strings attached

But the **`open-ce-builder`** is all you need to

- Build packages yourself
- Install and run your packages in a container



1. Install the open-ce builder
   `conda install -c open-ce open-ce-builder`
2. Decide on the packages you want
   - Individual packages (TF, XGBoost,..) or a complete environment with all OpenCE libraries
   - Pick the matching environment file from https://github.com/open-ce/open-ce/tree/main/envs
3. Build the libraries in a container
   `open-ce build env` `--container_build --container_tool podman` **`pytorch-env`**
4. Create a container image with the libraries
   `open-ce build image` `--conda_env_file=open-ce-`**`pytorch-env`**`.yaml --container_tool podman`

We’d like to hear from you

OpenCE currently provides recipes for **over 100** data science libraries.

You’re missing a library? Open a *feedstock request* in our GitHub repo!

- We’ll get back to you within a week

Libraries we like to include

- Not already provided by Anaconda
- Frequent releases, active community