

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

---

Suyog Jadhav  
Rocket Software Senior Manager  
[sujadh23@in.ibm.com](mailto:sujadh23@in.ibm.com)

Rajalakshmi Srinivasaraghavan  
Linux on Power Toolchain  
[rajalakshmi.srinivasaraghavan@ibm.com](mailto:rajalakshmi.srinivasaraghavan@ibm.com)

Alexander Lang  
Architect  
[alexlang@de.ibm.com](mailto:alexlang@de.ibm.com)

Si Win  
Data and AI on Power, PM  
[stwin@us.ibm.com](mailto: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

The screenshot shows the RocketCE for Power forum page. The header includes the Rocket software logo and navigation links like Home, Forums, Browse, Members, Getting Started, My Profile, Join the Forum, and Stage Entrance. A search bar is present. The main content area features a banner for "RocketCE for Power" and a section for "LATEST DISCUSSION POSTS" with a "POST A MESSAGE" button. Below this, there are two discussion posts: "MPI dependency" by Jamie Finney and "Releasing RocketCE Packages with OpenCE v1.91" by Rishabh Mishra. A "Volunteer Opportunities List" is also visible at the bottom.

The screenshot shows the Anaconda.org rocketce/packages page. The header includes the ANACONDA.ORG logo and a search bar. The main content area displays a list of packages under the "rocketce / packages" section. The packages are listed in a table with columns for Package Name, Access, Summary, and Updated. The packages include python, libpython-static, black, bazel-toolchain, bazel, av, arrow-cpp-proc, arrow-cpp, array-record, apache-beam, absl-py, ml\_dtypes, maturin, mamba, magma, and llvm-openmp.

Package Name	Access	Summary	Updated
python	public	General purpose programming language	2023-07-13
libpython-static	public	General purpose programming language	2023-07-13
black	public	The uncompromising code formatter.	2023-07-06
bazel-toolchain	public	Helper script to generate a crosscompile toolchain for Bazel with the currently activated compiler settings.	2023-07-06
bazel	public	build system originally authored by Google	2023-07-06
av	public	Pythonic bindings for Ffmpeg.	2023-07-06
arrow-cpp-proc	public	A meta-package to select Arrow build variant	2023-07-06
arrow-cpp	public	C++ libraries for Apache Arrow	2023-07-06
array-record	public	A new file format derived from Riegeli	2023-07-06
apache-beam	public	Apache Beam: An advanced unified programming model	2023-07-06
absl-py	public	Abseil Python Common Libraries, see https://github.com/abseil/abseil-py.	2023-07-06
ml_dtypes	public	A stand-alone implementation of several NumPy dtype extensions used in machine learning	2023-07-06
maturin	public	Build and publish crates with pyo3, rust-cpython and cffi bindings as well as rust binaries as python packages	2023-07-06
mamba	public	A fast drop-in alternative to conda, using libsolve for dependency resolution	2023-07-06
magma	public	Dense linear algebra library similar to LAPACK but for heterogeneous/hybrid architectures	2023-07-06
llvm-openmp	public	The OpenMP API supports multi-platform shared-memory parallel programming in C/C++ and Fortran.	2023-07-06

The screenshot shows the Red Hat Quayio Repositories page. The header includes the RED HAT Quayio logo and navigation links like EXPLORE, REPOSITORIES, and TUTORIAL. The main content area displays a list of repositories under the "Repositories" section. The repositories are listed in a table with columns for Repository Name, Last Modified, Activity, and Star. The repositories include rocketce / pytorch-cpu, rocketce / tensorflow-cpu, rocketce / tensorflow, and rocketce / pytorch. A sidebar on the right shows "Users and Organizations" with links to jadhasuyog, rocketce, and a "Create New Organization" button.

Repository Name	Last Modified	Activity	Star
rocketce / pytorch-cpu	04/20/2023		
rocketce / tensorflow-cpu	04/20/2023		
rocketce / tensorflow	04/20/2023		
rocketce / pytorch	04/20/2023		

# 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
<code>__vector_quad</code>	Accumulator data type.
<code>__builtin_mma_xxsetaccz()</code>	Reset accumulators.
<code>__builtin_mma_build_acc()</code> <code>__builtin_mma_disassemble_acc()</code>	Merge/Disassemble accumulators
<code>__builtin_mma_xvf*ger*()</code>	All precision Matrix multiply or multiply accumulate or negate-accumulate.
<code>__builtin_mma_pxvf*ger*()</code>	Prefixed/Masked matrix multiply operations.
<code>__builtin_vsx_build_pair()</code> <code>__builtin_vsx_disassemble_pair()</code>	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

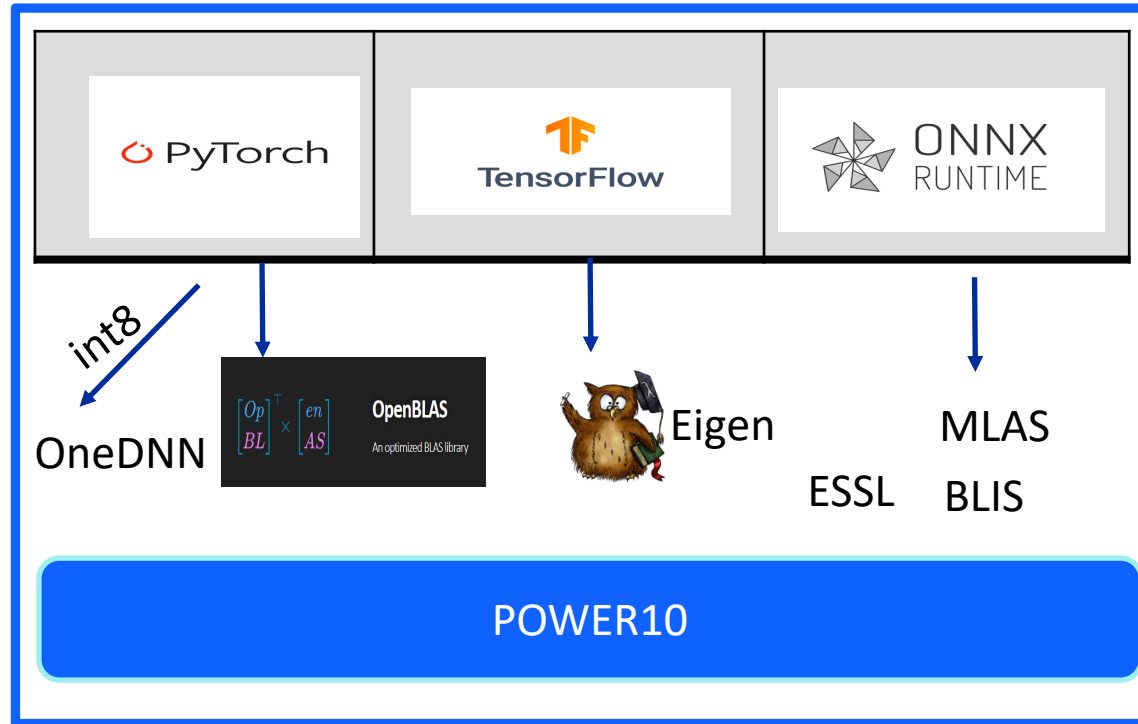
```
void
foo (vec_t *A, vec_t *B, vec_t *C)
{
    __vector_quad acc0, acc1;
    vector unsigned char result[4];

    __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
<b>OpenBLAS</b> (Used in PyTorch, Numpy) <a href="https://github.com/xianyi/OpenBLAS/">https://github.com/xianyi/OpenBLAS/</a>	0.3.13 and above	<b>MMA Level 3</b> GEMM functions optimized: <ul style="list-style-type: none"> <li>Sgemm [float]</li> <li>Dgemm [double]</li> <li>Cgemm [complex float]</li> <li>Zgemm [complex double]</li> <li>Sbgemm (BFloat16)</li> </ul> <b>Level 2</b> GEMV functions optimized for double type. <ul style="list-style-type: none"> <li>dgemv optimized to use power10 vector pair instructions.</li> <li>Zgemv optimized using MMA</li> </ul> <b>Level 1 vector-vector</b> functions optimized to use power10 vector pair instructions.
<b>Eigen</b> (Used by Tensorflow) <a href="https://gitlab.com/libeigen/eigen/">https://gitlab.com/libeigen/eigen/</a>	3.4	<b>MMA Level 3</b> GEMM (matrix-matrix) functions optimized for <ul style="list-style-type: none"> <li>Real float and double</li> <li>complex float and double</li> <li>bfloat16</li> </ul> <b>MMA &amp; VSX Level 2</b> GEMV (matrix-vector) functions optimized <ul style="list-style-type: none"> <li>Real float and double</li> <li>complex float and double</li> </ul>
<b>ONNXRuntime</b> <a href="https://github.com/microsoft/onnxruntime">https://github.com/microsoft/onnxruntime</a>	1.9.0 and above	<b>MMA Level 3</b> GEMM functions optimized. <ul style="list-style-type: none"> <li>Sgemm</li> <li>Dgemm in ORT 1.10</li> <li>Low precision: Int8 GEMM in 1.11</li> </ul>
Numpy <a href="https://github.com/numpy/numpy/">https://github.com/numpy/numpy/</a>	1.23.0	<ul style="list-style-type: none"> <li>MMA for GEMM comes through OpenBLAS</li> <li>Integer logical operation – and, or, not</li> <li>Integer arithmetic operation – floor, fmod, divide, remainder</li> <li>Integer comparison operation - greater, less than, equal</li> </ul>
OneDNN <a href="https://github.com/oneapi-src/oneDNN/">https://github.com/oneapi-src/oneDNN/</a>	2.7	<ul style="list-style-type: none"> <li>Low precision: Int8 GEMM with MMA (from 2.7 version)</li> <li>MMA for bf16 , float and double comes from OpenBLAS</li> </ul>
<b>BLIS</b> <a href="https://github.com/flame/blis">https://github.com/flame/blis</a>	0.9.0	<b>MMA Level 3</b> GEMM functions optimized. –Sgemm & Dgemm Low precision GEMM functions introduced and optimized.[sandbox] <ul style="list-style-type: none"> <li>Bfloat16 / float16</li> <li>Int16 / int8 /int4</li> </ul>

- 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

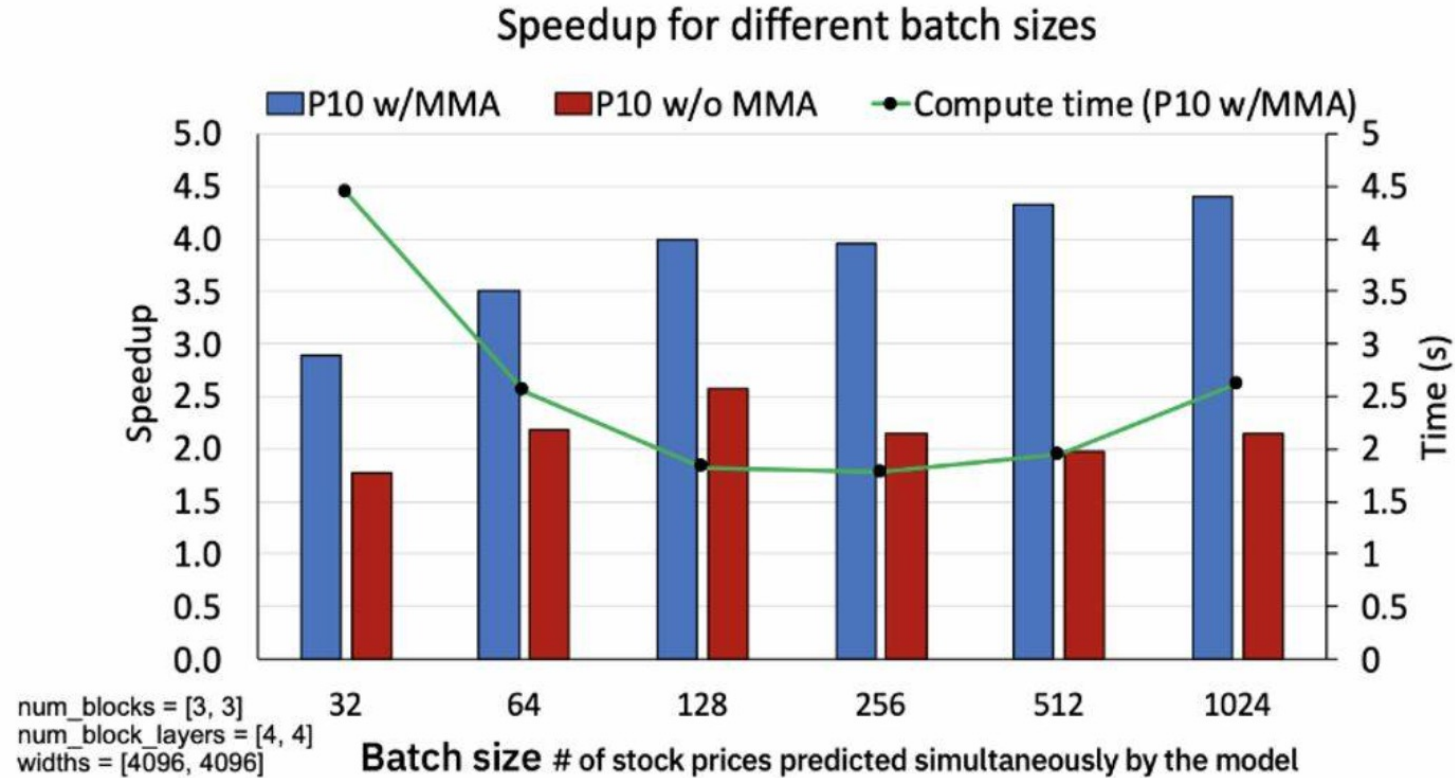
CPU only packages enabled with P10 MMA for Python 3.8 & above are available at:

<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



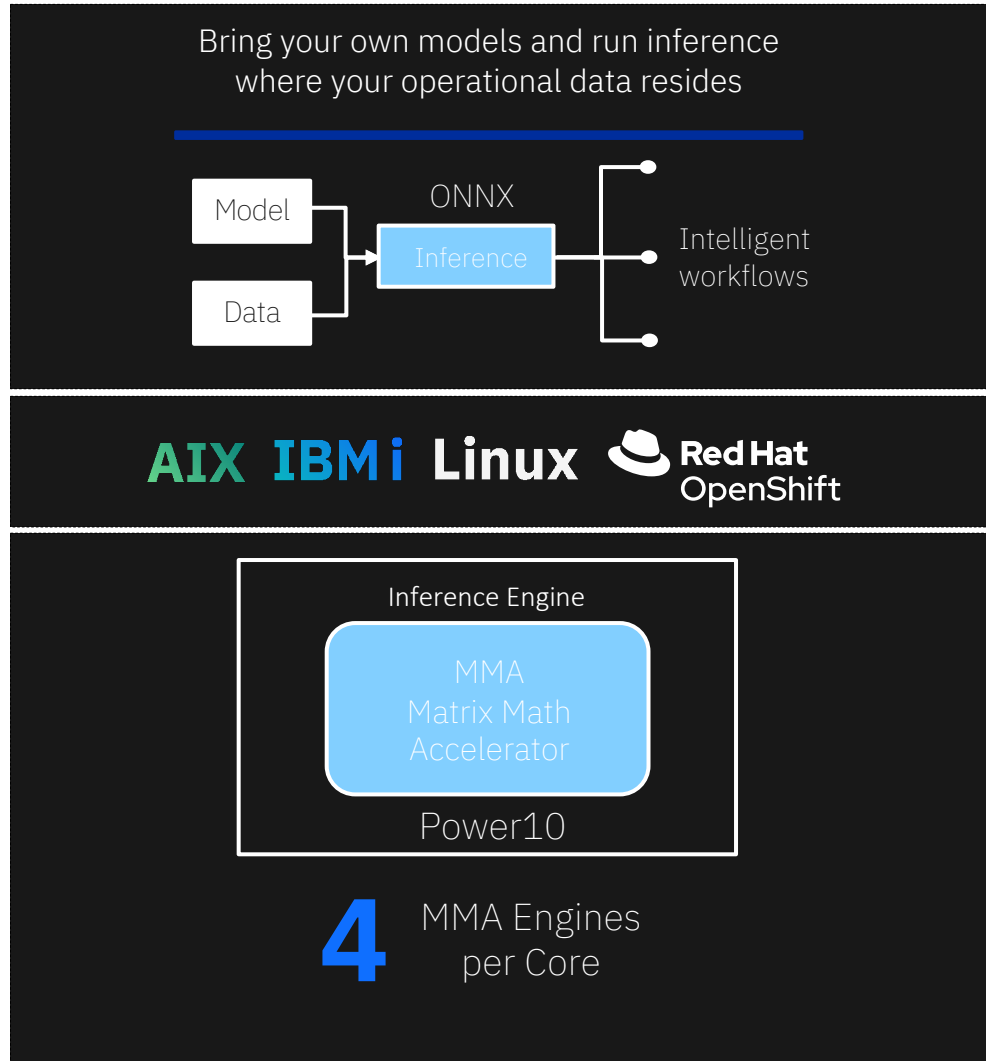
N-BEATS\* model from pytorch\_forecasting. Yahoo dataset.

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>

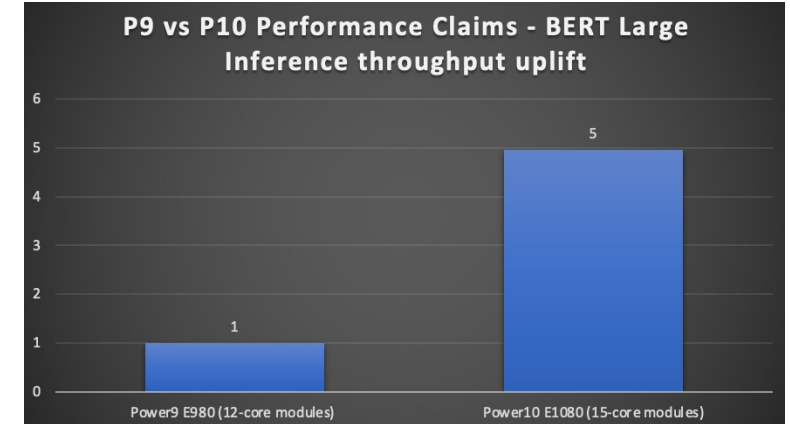
Reference : <https://developer.ibm.com/tutorials/power10-business-inferencing-at-scale-with-mma/>

# 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

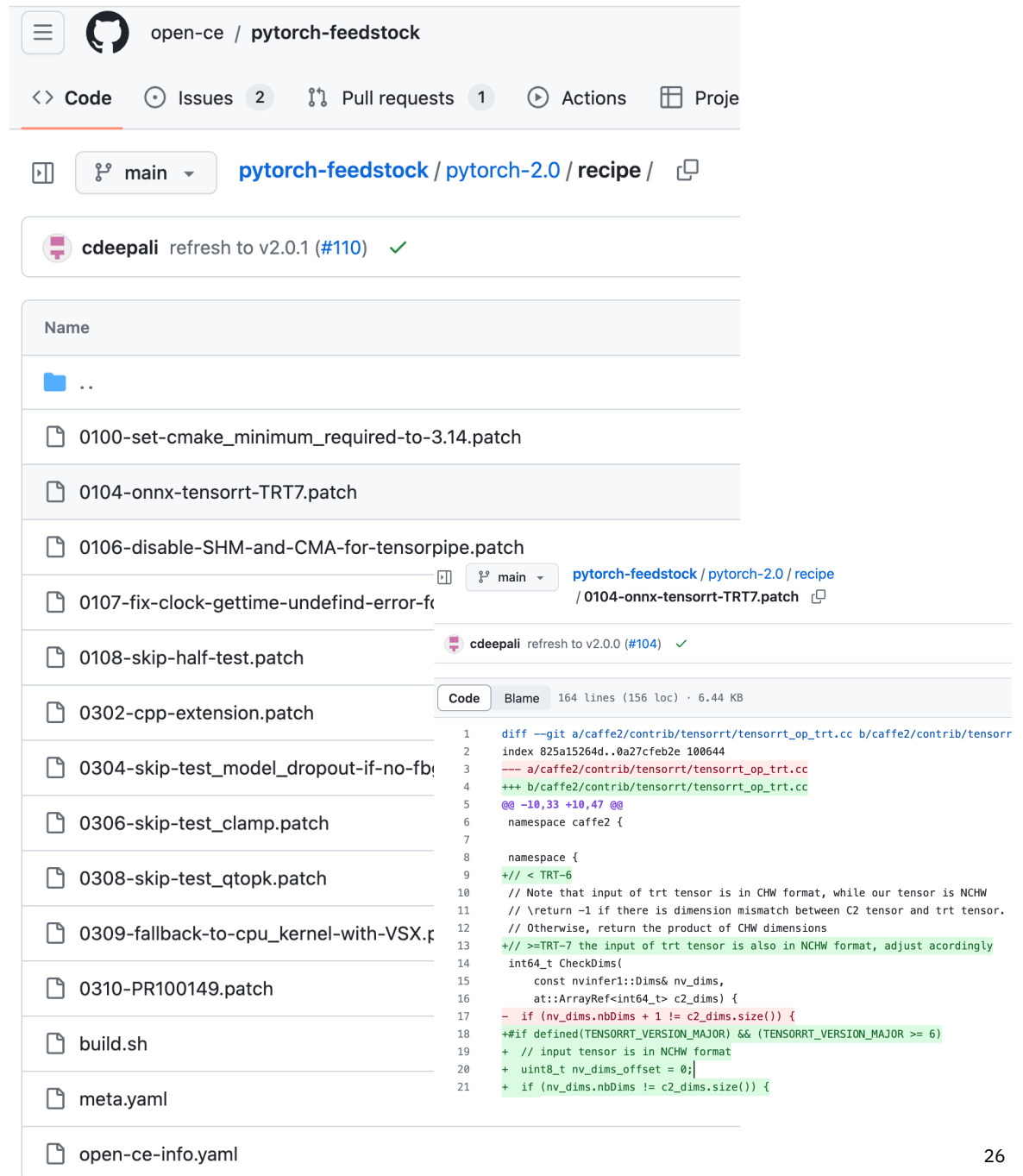
The cookbook for AI on Power

# 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>



The screenshot shows the GitHub repository 'open-ce / pytorch-feedstock'. The repository has 2 issues and 1 pull request. The 'recipe' folder is selected, showing a list of files including patch files and a build.sh script. A diff view is shown for the file '0104-onnx-tensorrt-TRT7.patch'.

Files in the recipe folder:

- 0100-set-cmake\_minimum\_required-to-3.14.patch
- 0104-onnx-tensorrt-TRT7.patch
- 0106-disable-SHM-and-CMA-for-tensorpipe.patch
- 0107-fix-clock-gettime-undefind-error-fr
- 0108-skip-half-test.patch
- 0302-cpp-extension.patch
- 0304-skip-test\_model\_dropout-if-no-fb
- 0306-skip-test\_clamp.patch
- 0308-skip-test\_qtopk.patch
- 0309-fallback-to-cpu\_kernel-with-VSX.p
- 0310-PR100149.patch
- build.sh
- meta.yaml
- open-ce-info.yaml

Diff view for 0104-onnx-tensorrt-TRT7.patch:

```
1 diff --git a/caffe2/contrib/tensorrt/tensorrt_op_trt.cc b/caffe2/contrib/tensorrt
2 index 825a15264d..0a27cfeb2e 100644
3 --- a/caffe2/contrib/tensorrt/tensorrt_op_trt.cc
4 +++ b/caffe2/contrib/tensorrt/tensorrt_op_trt.cc
5 @@ -10,33 +10,47 @@
6 namespace caffe2 {
7
8 namespace {
9 +// < TRT-6
10 // Note that input of trt tensor is in CHW format, while our tensor is NCHW
11 // \return -1 if there is dimension mismatch between C2 tensor and trt tensor.
12 // Otherwise, return the product of CHW dimensions
13 +// >=TRT-7 the input of trt tensor is also in NCHW format, adjust accordingly
14 int64_t CheckDims(
15     const nvInfer1::Dims& nv_dims,
16     at::ArrayRef<int64_t> c2_dims) {
17 - if (nv_dims.nbDims + 1 != c2_dims.size()) {
18 + if (nv_dims.nbDims + 1 != c2_dims.size()) {
19 + #if defined(TENSORRT_VERSION_MAJOR) && (TENSORRT_VERSION_MAJOR >= 6)
20 + // input tensor is in NCHW format
21 + uint8_t nv_dims_offset = 0;
22 + if (nv_dims.nbDims != c2_dims.size()) {
```

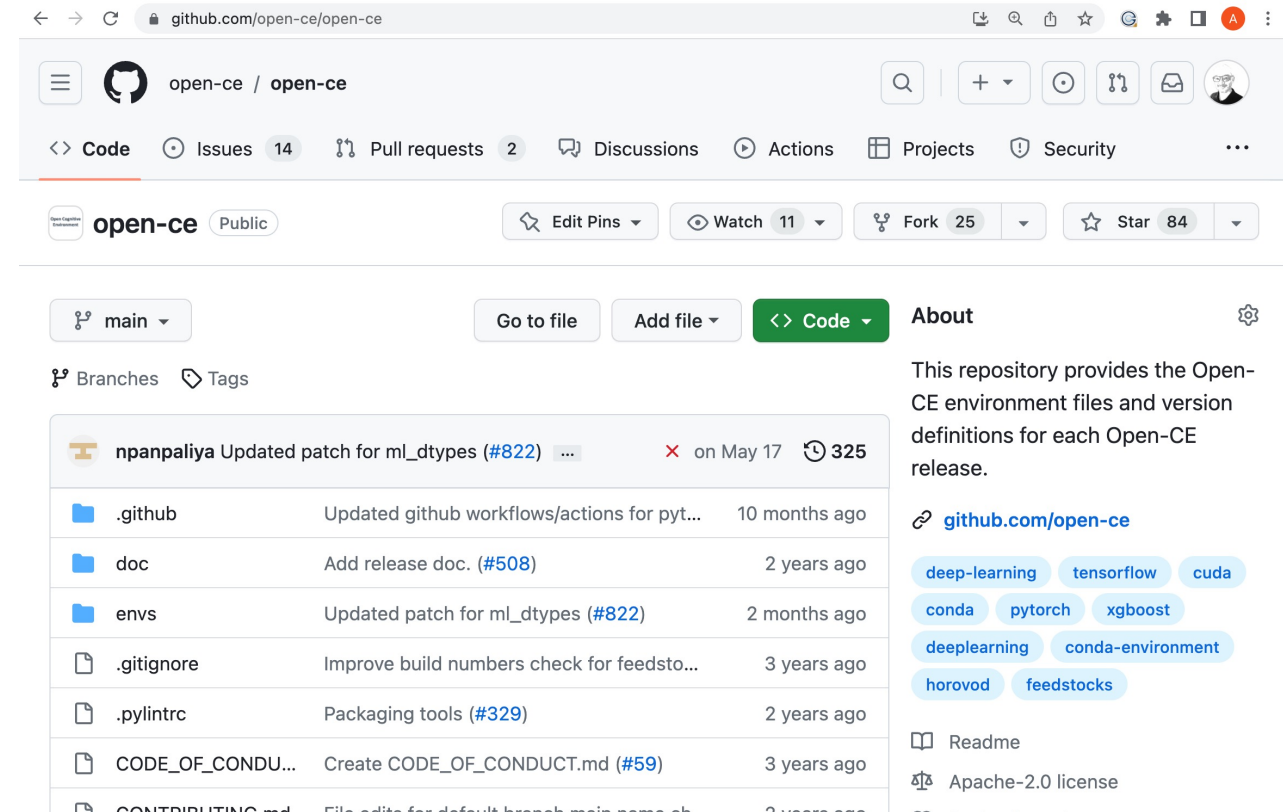
# IBM OpenCE: optimized conda build recipes for Power <https://github.com/open-ce/open-ce>

**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 `open-ce` directory. These files 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

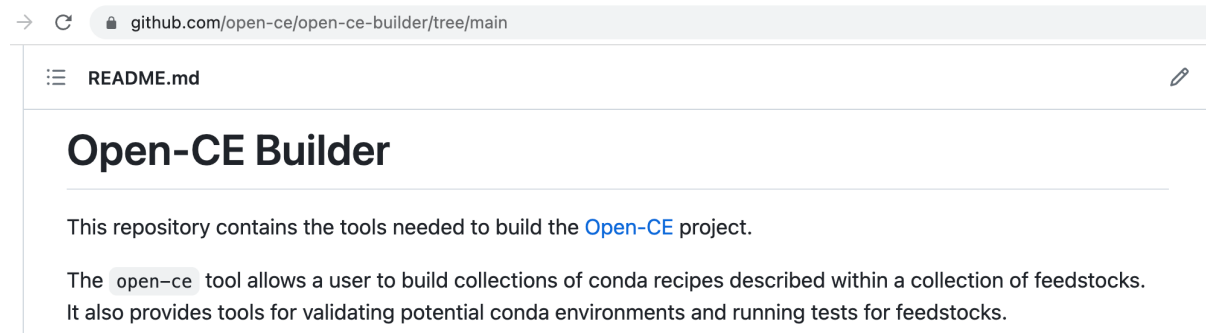
# Build it yourself

<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

