DECADES Programmer Guide

Hello and welcome to the DECADES Programmer Guides!

We're here to give you some guidance on efficiently programming and evaluating your programs on our new architecture, using our specialized language infastructures.

Note

This documentation is provided in jupyter notebooks, but not all notebooks are executable from start to finish. For one, they require the installation of all our specialized tools. We've provided these in a docker image for you, but unfortuantly we could not link it to the notebook.

Instead these notebooks are intended as stylized READMEs. This is useful as our programming interfaces are largely in Python. All code in code blocks is executable in the right context (i.e. in the docker container).

Because you are running our docker container on an x86 platform, your results will be provided through emulation on x86 and simulation. This guide will help you rapidly develop in emulation mode and switch to heavy-weight simulation when necessary.

Choose your own adventure...

The workloads that the DECADES framework aims to accelerate come in two flavors:

- Neural Networks (Training and Inference) (DECADES_TensorFlow.ipynb)
- Graph Analytics (kernel_implementation.ipynb)

So we've provided two distinct, but natural paths for programming each type of workload. For NN training and inference, we've provided a large set of TensorFlow functions you can use. For graph analytics, we've provided a more general Python interface through Numba.

Both of these tools are quite popular in recent years and if you've used any of them before, then you've got a great start!

We'll briefly describe each path, link to their detailed documentation and then describe a general evaluation framework we've provided.

Programming DECADES with TensorFlow

TensorFlow (https://www.tensorflow.org) is a popular framework for programming neural network tasks. If your work load (i.e. a DNN application) uses a lot of tensor operations (activations, matrix multiplication, convolution), then this is the path for you!

At a high level, programming with the DECADES platform through TensorFlow consists of writing your neural network using the library functions and passing our framework the computation graph. We then analyze the graph and simulate an execution of one batch of training (or inference) and report the results.

Our DECADES infastructure can accelerate such workloads using a range of specialized accelerators. Details can be found here (http://decades.cs.princeton.edu/).

Programming DECADES with Numba

Numba (https://numba.pydata.org) is a popular framework for writing high performance python code. The secret is that Numba compiles computationally heavy functions into [LLVM] (https://llvm.org) (https://llvm.org)), which can then be executed natively on the machine. If your workload is a graph analytics application; or simply an application that cannot easily be mapped to tensor operations, then this is the path for you!

Our DECADES compiler infastructure is based on LLVM, so this is a natural fit for us to target!

In this flow, you will write a single computation kernel using

- numpy (https://www.numpy.org) data structures and functions

- our custom DECADES library, which mimics the scipy (https://www.scipy.org) package
- multiple threads (or tiles in DECADES parlance)

This kernel is then compiled and launched through our framework.

Our DECADES infastructure can accelerate such workloads using parallelism and a specialized latency hiding technique called *decoupled supply/compue*. More details can be found here (http://mrmgroup.cs.princeton.edu/papers/taejun_micro15.p

Evaluation

Once you have written your kernel through one of the two flows above, it is time to evaluate it! We've provided some nice programs to help you do this. These programs will take your kernel and report execution metrics for:

1. A baseline application/machine (e.g. as provided by an evaluation)
2. Your application simulated on our projected future DECADES chip

You can easily see speedups (or slowdowns if you aren't careful!).

We can report on several metrics, but the default metric (and the one asked for in the project) is GOPs/Watt or Giga-operations per Watt.

In case you are interested in other metrics, we can provide them as well. For these evaluation tools, please see the documentation here (evaluation_guide.ipynb).

Lower Level Documentation

We hope that users will mostly use our interfaces documented in this guide. However, our lower level tools are available and documented.

Our tools are all available on the docker.

Our lower level tools (e.g. C++ compiler, simulator) are documented here (DECADES_Doc_and_Specs.pdf).

Summary

DECADES can be programmed through two distinct, but natural paths: 1) Python TensorFlow or 2) Python Numba.

These paths are a high-level interface to our lower level tools, which are also documented.

In most cases, users do not need to be exposed to lower levels, and can perform rigorous evaluations just using our framework documented here.

All tools are provided in a docker image.

Have fun!