Customer Enablement of Habana® Gaudi® Amazon EC2 DL1 Instances

September 2021

habana®
An Intel Company

Contents

1. Introduction

Demand for high-performance Deep Learning (DL) training compute is accelerating with the growing number of applications and services based on image and gesture recognition in videos, speech recognition, natural language processing, recommendation systems and more. With this increased demand comes the need for greater training speed, throughput and capacity, which translate into the growing need for efficient scaling of training systems. The Habana® Gaudi® processor is designed to maximize training throughput and efficiency, while providing developers with optimized software and tools that scale to many workloads and systems. Habana Gaudi software was developed with the end-user in mind, providing versatility and ease of programming to address the unique needs of users' proprietary models, while allowing for a simple and seamless transition of their existing models over to Gaudi.

This document provides an overview of Habana Gaudi architecture, SynapseAI® software suite and the Habana Developer Platform. Section 2 provides background on the Gaudi processor technology, Section 3 presents the SynapseAI Software Suite, Section 4 focuses on the Habana Developer Platform and Section 5 on enabling users migrate to Gaudi. Section 6 provides details on the Gaudi-based EC2 instances. The Appendix contains brief information on the in-premise solutions.

2. Gaudi Architecture

Gaudi has been designed from the ground up for accelerating DL training workloads. Its heterogeneous architecture comprises a cluster of fully programmable Tensor Processing Cores (TPC) along with its associated development tools and libraries, and a configurable Matrix Math engine.

The TPC core is a VLIW SIMD processor with instruction set and hardware that were tailored to serve training workloads efficiently. It is programmable, providing the user with maximum flexibility to innovate, coupled with many workload-oriented features, such as:

- GEMM operation acceleration
- Tensor addressing
- Latency hiding capabilities
- Random number generation
- Advanced implementation of special functions

The TPC core natively supports the following data types: FP32, BF16, INT32, INT16, INT8, UINT32, UINT16 and UINT8. The Gaudi memory architecture includes on-die SRAM and local memories in each TPC. In addition, the chip package integrates four HBM devices, providing 32 GB of capacity and 1 TB/s bandwidth. The PCIe interface provides a host interface and supports both generation 3.0 and 4.0 modes.

Gaudi is the first DL training processor that has integrated RDMA over Converged Ethernet (RoCE v2) engines on-chip. With bi-directional throughput of up to 2 Tb/s, these engines play a critical role in the inter-processor communication needed during the training process. This native integration of RoCE allows customers to use the same scaling technology, both inside the server and rack (termed as scale-up), as well as to scale across racks (scale-out). These can be connected directly between Gaudi processors, or through any number of standard Ethernet switches.

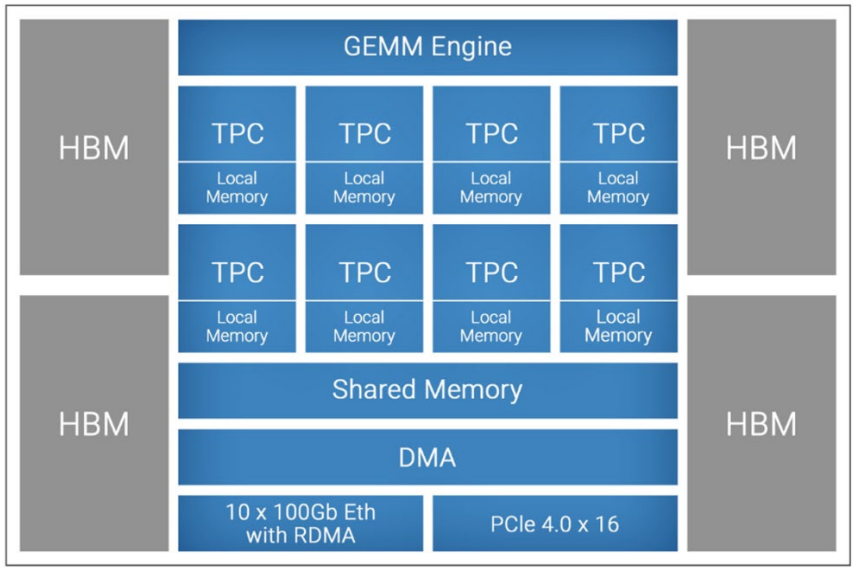

Figure 1: Gaudi Processor High-level Architecture

3. SynapseAI® Software Suite

Designed to facilitate high-performance DL training on Habana's Gaudi accelerators, SynapseAI Software Suite enables efficient mapping of neural network topologies onto Gaudi hardware. The software suite includes Habana's graph compiler and runtime, TPC kernel library, firmware and drivers, and developer tools such as the TPC SDK for custom kernel development and SynapseAI Profiler. SynapseAI is integrated with the popular frameworks, TensorFlow and PyTorch, and performance-optimized for Gaudi. Figure 2 shows a pictorial view of the SynapseAI software suite.

3.1. Graph Compiler and Runtime

The SynapseAI graph compiler generates optimized binary code that implements the given model topology on Gaudi. It performs operator fusion, data layout management, parallelization, pipelining and memory management, and graph-level optimizations. The graph compiler uses the rich TPC kernel library, which contains a wide variety of operations (for example, elementwise, non-linear, non-GEMM operators). Kernels for training have two implementations, forward and backward.

Given the heterogenous nature of Gaudi hardware (Matrix Math engine, TPC and DMA), the SynapseAI graph compiler enables effective utilization through parallel and pipelined execution of framework graphs. SynapseAI uses stream architecture to manage concurrent execution of asynchronous tasks. It includes multi-stream execution environment, supporting Gaudi's unique combination of compute and networking, exposing a multi-stream architecture to the framework. Streams of different types — compute, networking and DMA — are synchronized with one another at high performance and with low run-time overheads.

3.2. Habana Communication Libraries

The Habana Communication Library enables efficient scale-up communication between Gaudi processors within a single node and scale-out across nodes for distributed training, leveraging Gaudi's high performance RDMA communication capabilities. It has an MPI look-and-feel and supports point-to- point operations (for example, Write, Send) and collective operations (for example, AllReduce, AlltoAll) that are

performance -optimized for Gaudi. Habana Collective Communications Library (HCCL) that is Habana's implementation of standard collective communication routines with NCCL-compatible API.

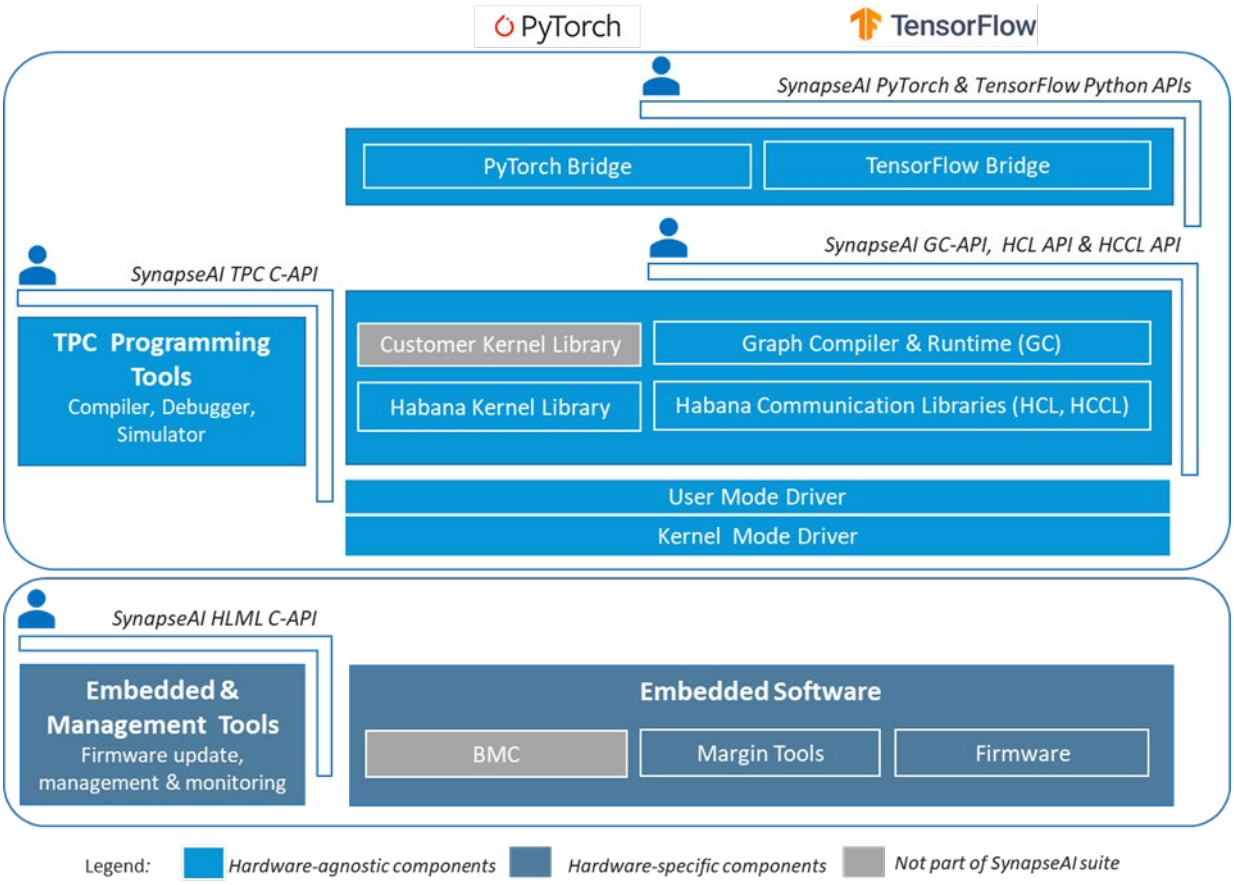

Figure 2: SynapseAI Software Suite

3.3. TPC Programming

The SynapseAI TPC SDK includes an LLVM-based TPC-C compiler, a simulator and debugger. These tools facilitate the development of custom TPC kernels, and we have used this very SDK to build the high-performance kernels provided by Habana. Users can thereby develop customized deep learning models and algorithms on Gaudi to innovate and optimize to their unique requirements.

The TPC programming language, TPC-C, is a derivative of C99 with added language data types to enable easy utilization of processor-unique SIMD capabilities. It natively supports wide vector data types to assist with programming of the SIMD engine (for example, float64, uchar256 and so on). It has many built-in instructions for deep learning, including:

- Tensor-based memory accesses
- Accelerations for special functions
- Random number generation
- Multiple data types

A TPC program consists of two parts – TPC execution code and host glue code. TPC code is the ISA executed by the TPC processor. Host code is executed on the host machine and provides specifications regarding how the program input/outputs can be dynamically partitioned between the numerous TPC processors in the Habana Gaudi device.

3.4. DL Framework Integration

Popular DL frameworks such as TensorFlow and PyTorch are integrated with SynapseAI and optimized for Gaudi. This section provides a brief overview of the SynapseAI TensorFlow integration. It illustrates how SynapseAI does much of the mapping and optimization under the hood, while customers still enjoy the same abstraction, they are accustomed to today.

The SynapseAI TensorFlow bridge receives a computational graph of the model from the TensorFlow framework and identifies the subset of the graph that can be accelerated by Gaudi. These subgraphs are encapsulated and executed optimally on Gaudi. Figure 3 shows an example of encapsulation performed on the TensorFlow framework graph. The yellow node is not supported on Gaudi, while blue nodes can execute on Gaudi. Subgraphs with blue nodes are identified and encapsulated. The original graph is modified to replace the subgraphs with their corresponding encapsulated nodes.

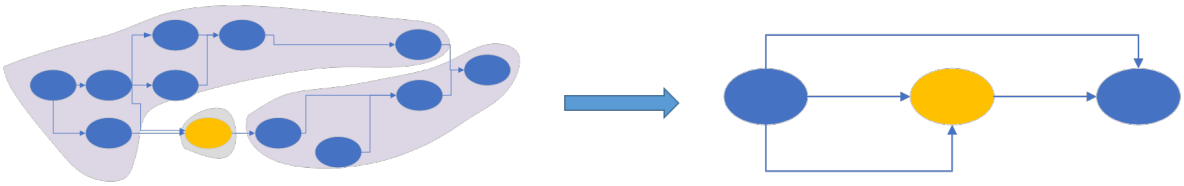

Figure 3. Subgraph selection and encapsulation in TensorFlow framework graph

The framework runtime then executes the modified graph. Per node, a corresponding SynapseAI graph is created and compiled. For performance optimization, the compilation recipe is cached for future use. After allocating memory, the recipe is enqueued for execution on a SynapseAI stream.

3.5. Embedded Software and Tools

The SynapseAI software suite include embedded software and tools intended primarily for server developers and IT personnel who manage server deployments. As these are not relevant for EC2 users, we have not included them in this document.

4. Habana Developer Platform

Developer.habana.ai is the hub for Habana developers from where they will find Gaudi software, optimized models, documentation and so on. Please read our blog "Introducing the Habana Developer Site".

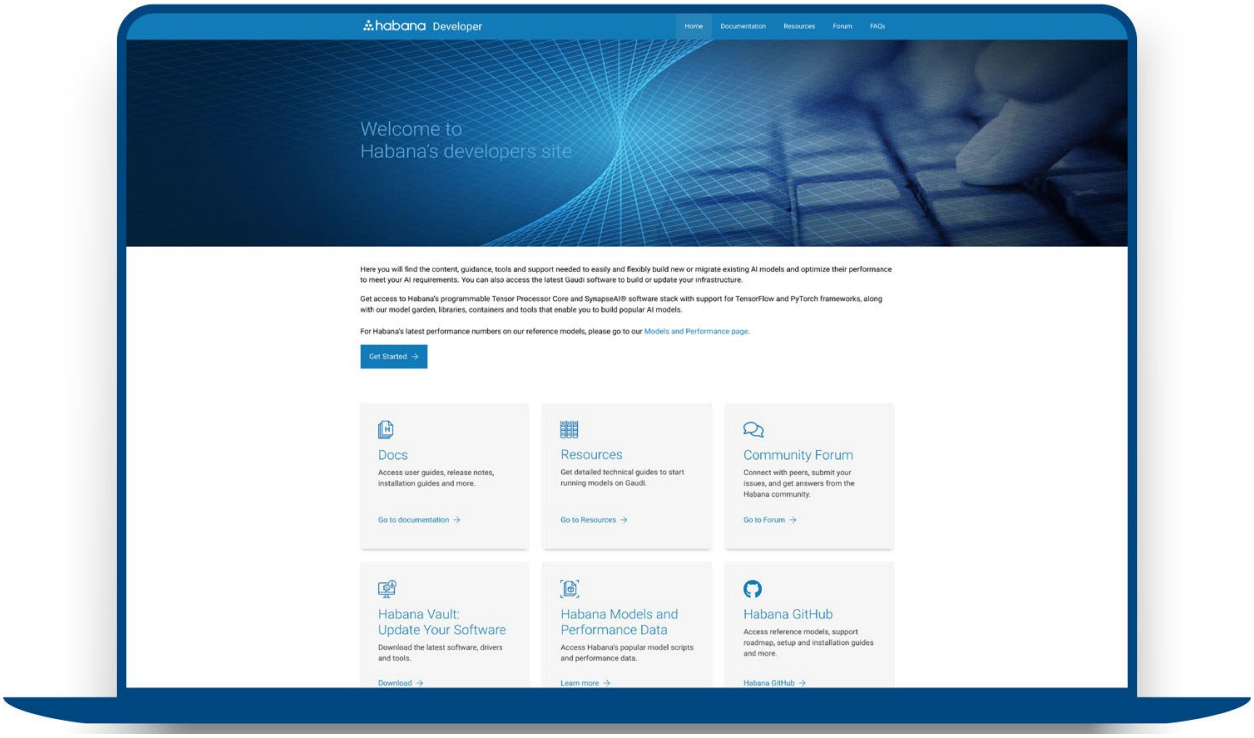

Figure 4: High-level Structure of Habana Developer Platform

Habana GitHub contains repositories open to the general public. Section 4.2 provides details on some of the repositories available. GitHub will be the primary channel for developer support, where Habana domain experts will actively engage in supporting developers and users. In addition, the Forum section on Developer Platform will provide developers with another source where they can find answers to questions and collaborate with the developer community within and outside of Habana.

The Documentation section will host detailed documentation on various software components, user guides and release notes. It is web-based and fully searchable content. It will also contain short video tutorials to assist end users in getting started and running models on Gaudi.

4.1. Vault and SynapseAI Container Registry

The Vault provides access to Habana hardware collateral and software releases. End users and the general community will have Open Access to all publicly available Habana content. Privileged Access will be provided for customers with business relationships that require NDA, such that they can download Habana content as well as upload their proprietary content for Habana implementations into their secured Account Vault, as needed. ODM/OEM partners will have NDA Access to download Habana content.

Containers can be deployed easily and consistently, regardless of whether the target environment is a private data center or the public cloud. The Gaudi-optimized DL frameworks containers are delivered with all necessary dependencies including the complete SynapseAI software.

SynapseAI is integrated and validated with officially released versions of TensorFlow and PyTorch. Support for framework operators, features and APIs will continue to evolve over time. We will update to

and support the latest versions of framework releases. Please refer to the Release Notes on docs.habana.ai for the supported versions.

The table below highlights supported versions as of September 2021:

Supported Frameworks	TensorFlow2 and PyTorch
Operating Systems	Ubuntu 18.04 and 20.04, AWS Linux2
Container Runtimes	Docker (minimum Docker CE version 18.09)
Distributed Training Schemes	TensorFlow with Horovod and tf.distribute PyTorch distributed (native)

The Vault contains publicly available official releases of SynapseAI TensorFlow and PyTorch Docker container images.

4.2. Habana GitHub

This section provides an overview of select repositories on Habana GitHub.

The Setup and Install repository contains instructions and guidance on setting up the environment with Gaudi firmware and drivers, installing SynapseAI TensorFlow and PyTorch containers, and running with containers. It also contains Dockerfiles and instructions to build your own Docker images for SynapseAI with TensorFlow or PyTorch.

The Model References repository contains examples of DL training models (primarily vision, natural language processing, recommendation systems) that are implemented on Habana Gaudi. Each model comes with model scripts, recipes and instructions to run the models. As of September 2021, we have 20 models available on the GitHub. The tables below show the performance results. This information is posted on our developer site and updated for every release.

TensorFlow Reference Models Performance*

Framework	Model	#HPU	Precision	Time to Train	Accuracy	Throughput	Batch Size
TensorFlow 2.5.1	ResNet50 Keras LARS	1	Mixed	8h36m	76.09	1700 images/sec	256
TensorFlow 2.5.1	ResNet50 Keras LARS (with horovod)	8	Mixed	1h11m	76.13	12200 images/sec	256
TensorFlow 2.5.1	ResNet50 Keras LARS (with tf distribute)	8	Mixed	1h 9min	75.96	12900 images/sec	256
TensorFlow 2.5.1	ResNet50 Keras SGD	1	Mixed	19h 31min	76.2	1700 images/sec	256
TensorFlow 2.5.1	ResNet50 Keras SGD	8	Mixed	2h 39min	76.3	12580 images/sec	256
TensorFlow 2.5.1	ResNet50 Keras SGD	16	Mixed	43min	75.55	23900 images/sec	256
TensorFlow 2.5.1	ResNet50 Keras SGD	32	Mixed	24min	75.97	46700 images/sec	256

TensorFlow 2.6.0	ResNext101	1	Mixed		79.07	663 images/sec	128
TensorFlow 2.6.0	ResNext101	8	Mixed	6h 56min	79.15	4780 images/sec	128
TensorFlow 2.5.1	SSD ResNet34	1	Mixed	3h 35min	22.97	470 images/sec	128
TensorFlow 2.5.1	SSD ResNet34	8	Mixed	35min	22.04	3406 images/sec	128
TensorFlow 2.6.0	Mask R-CNN	1	Mixed	25h 18min	33.99	15 images/sec	4
TensorFlow 2.6.0	Mask R-CNN	8	Mixed	4h 31min	34.23	99 images/sec	4
TensorFlow 2.5.1	Unet2D	1	Mixed	20min	88.79	48 images/sec	8
TensorFlow 2.5.1	Unet2D	8	Mixed	7min	88.09	360 images/sec	8
TensorFlow 2.5.1	Unet3D	1	Mixed	1h 47min	88.96	5.2 images/sec	2
TensorFlow 2.5.1	Unet3D	8	Mixed	19min	89.06	35 images/sec	2
TensorFlow 2.5.1	DenseNet (with tf.distribute)	8	Mixed	5h 15min	73.44	5423 images/sec	128
TensorFlow 2.5.1	RetinaNet	1	fp32	8h 53min	27.35	12 images/sec	8
TensorFlow 2.6.0	BERT-Large Fine Tuning (SQUAD)	1	Mixed	1h 8m	92.91	52 sentences/sec	24
TensorFlow 2.6.0	BERT-Large Fine Tuning (SQUAD)	8	Mixed	22min	93.26	391 sentences/sec	24
TensorFlow 2.6.0	BERT-Large Pre Training	1	Mixed			Phase 1 165 sps Phase 2 258 sps	Phase 1 – 64 Phase 2 – 8
TensorFlow 2.6.0	BERT-Large Pre Training	8	Mixed			Phase 1 1310sps Phase 2 249 sps	Phase 1 – 64 Phase 2 – 8
TensorFlow 2.6.0	BERT-Large Pre Training	32	Mixed	39h	Phase 1 loss 1.12 Phase 2 loss 0.86	Phase 1 – 5400sps Phase 2 – 1030sps	Phase 1 – 64 Phase 2 – 8
TensorFlow 2.6.0	Transformer	8	Mixed	17h 43min	26.5	154020	4096
TensorFlow 2.5.1	T5-base Fine Tuning	1	Mixed	16min	94.1	115	16
TensorFlow 2.5.1	Albert-Large Fine Tuning (SQUAD)	8	Mixed	14min 42s	F1 90.9 EM 84.18	436 sentences/sec	32
TensorFlow 2.5.1	Albert-Large Pre Training	1	Mixed			Phase 1 177sps Phase 2 36sps	Phase 1 – 64 Phase 2 – 8
TensorFlow 2.5.1	EfficentDet	8	fp32	4days 22h	33.8	91.4 images/sec	8

TensorFlow 2.5.1	CycleGan	1	Mixed	9h 25min		5.9	2
TensorFlow 2.5.1	CycleGan	8	Mixed	9h 40min		44	2
TensorFlow 2.5.1	SegNet	1	Mixed	8.5min	89.57	303 images/sec	16
TensorFlow 2.5.1	SegNet	4	Mixed	3.9min	90.6	104 ages/sec	16

PyTorch Reference Models Performance*

Framework	Model	#HPU	Precision	1.0 TTT	1.0 Accuracy	1.0 Throughput	Batch Size
PyTorch 1.8.1	ResNet50	1	Mixed		76.04	1583 images/sec	256
PyTorch 1.8.2	ResNet50	8	Mixed	5h 37min	75.95	7350 images/sec	256
PyTorch 1.8.2	ResNet50	16	Mixed	3h 54min	75.86	12600 images/sec	256
PyTorch 1.8.2	ResNext101	1	Mixed		N/A	725 images/sec	128
PyTorch 1.8.2	ResNext101	8	Mixed	10h 50min	78.01	3780 images/sec	128
PyTorch 1.8.2	BERT-Large Fine Tuning (SQUAD)	1	Mixed	1h 11min	93.3	46 sentences/sec	24
PyTorch 1.8.2	BERT-Large Fine Tuning (SQUAD)	8	Mixed	30min	92.8	330 sentences/sec	24
PyTorch 1.8.2	BERT-Large Pre Training	1	Mixed			Phase 1 – 155 sentences/sec Phase 2 – 31 sentences/sec	64
PyTorch 1.8.2	BERT-Large Pre Training	8	Mixed			Phase 1 – 1230 sentences/sec Phase 2 – 245 sentences/sec	64
PyTorch 1.8.2	DLRM	1	Mixed			47086 queries/sec	512

***System Configuration:** HPU: Habana Gaudi® HL-205 Mezzanine cards, System: HLS-1 with eight HL-205 HPU and two Intel® Xeon® Platinum 8280 CPU @ 2.70GHz, and 756GB of System Memory, Software: Ubuntu20.04, SynapseAI Software version 1.0.1-81. Models run with Tensorflow v2.5.1 use this Docker image, and ones with v2.6.0 use this Docker image. Models run with PyTorch v1.8.2 use this Docker image. Environment: These workloads are run using the Docker images running directly on the Host OS.*

Performance varies by use, configuration and other factors. All information provided here is subject to change without notice. Habana Labs may make changes to its test conditions and internal reliability goals at any time. Contact your Habana Labs representative to obtain the latest Habana Labs product specifications and roadmaps. Your costs and results may vary.

We plan to expand our model coverage continuously and provide a wide variety of examples for users. In the process, we expect to broaden framework operator coverage. Our TPC kernel library is continually evolving and growing. Our roadmap repository will include updates on new models that we plan to support.

5. Migration to Gaudi

Switching from a familiar DL platform and workflow to a new one takes effort. Our goal is to minimize this effort and lower the barriers wherever possible. We expect most users will be able to take existing models with minor changes to existing scripts and run on Gaudi. Habana GitHub will contain migration guides and examples to assist users with porting their current models to run on Gaudi. In this section, the focus is on TensorFlow. A similar approach applies to PyTorch as well. More information on migrating models to Gaudi is available in the Migration Guide on docs.habana.ai.

The assumption is that the user is familiar with TensorFlow. The SynapseAI TensorFlow user guide will provide an overview of SynapseAI integration with TensorFlow, APIs and operators that are supported, and so on. The migration guide helps users develop a better understanding of how to port their current TensorFlow models to Gaudi and provide practical tips to assist in their effort. In this section, we show the minimum set of changes required to run a TensorFlow Keras model that does not contain any custom kernels.

```
import tensorflow as tf
from TensorFlow.common.library_loader import load_habana_module
load_habana_module()
(x_train, y_train), (x_test, y_test) = tf.keras.datasets.mnist.load_data()
x_train, x_test = x_train / 255.0, x_test / 255.0
model = tf.keras.models.Sequential([
      tf.keras.layers.Flatten(input_shape=(28, 28)),
      tf.keras.layers.Dense(10),
])
loss = tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True)
optimizer = tf.keras.optimizers.SGD(learning_rate=0.01)
model.compile(optimizer=optimizer, loss=loss, metrics=['accuracy'])
model.fit(x_train, y_train, epochs=5, batch_size=128)
model.evaluate(x_test, y_test)
```

The minimal changes to enable training on the Habana Gaudi device are highlighted in **bold:**

- On line 2, we import ‘load_habana_module’ needed to enable Gaudi
- On line 2, we now call the ‘load_habana_module()’ function to enable Gaudi.

Once loaded, the **HPU** device is registered in TensorFlow and prioritized over CPU. When an operator is available for both CPU and HPU, the operator is assigned to the HPU. When it is not supported on HPU, it runs on the CPU

The example code for training on multiple Gaudi devices in a single server as well as multi-node training (scale-out) on the Habana GitHub.

To enable developers get familiar with TPC programming for custom kernels, we have published the TPC programming user guide, training videos and sample code. Check out the Habana Developer site for additional details.

6. Getting Started with AWS DL1 Training Instances

Gaudi®-based Amazon EC2 DL1 Training Instances feature 8 Gaudi accelerators, AWS custom Intel Xeon Cascade Lake processors, 400 Gbps networking and 4TB of NVMe storage. These instances are available as standard EC2 instances with an easy to use, pay-as-you-go usage model and deliver up to 40% better price performance than GPU-based instances.**

Instance	Gaudi	vCPU	Memory	Networking
DL1.24xlarge	8	96	768	400Gbps

Developers will be able to benefit from full stack of Amazon EC2 services on the DL1 instances:

- AWS Deep Learning AMIs for Gaudi, on Ubuntu18.04 and Amazon Linux 2
- AWS Deep Learning Containers (DLC) for TensorFlow and PyTorch
- AWS ECS and EKS orchestration for containerized applications
- Efficient scaling across multiple Gaudi-based EC2 Instances

Integration with Amazon SageMaker is coming soon for users looking for a managed service for building and training machine learning applications.

Users can start with the pre-built AWS Deep Learning AMI (DLAMI) or AWS Deep Learning Containers (DLC) when launching a DL1 instance from EC2. In future, we will also publish Habana Gaudi Base AMIs on the AWS marketplace. Once the instance is launched, users can immediately get started with training their models on Gaudi. To migrate existing models or use one of Habana reference models or develop a model from scratch, users can refer to the content available on our Developer site. For questions, help requests and support on issues, users are invited to post on the Habana Forum or file issue tickets on the Habana GitHub repositories.

Please refer to the links below for information on getting started with a DL1 Training instances on EC2:

- Get started with Amazon EC2 DL1 Training Instances
- Amazon Elastic Container Service (Amazon ECS)
- Amazon Elastic Kubernetes Service (Amazon EKS)
- DLAMI release notes
- DLC Release notes

*** Legal disclaimer: The price performance claim is made by AWS and based on AWS internal testing. Habana Labs does not control or audit third-party data; your price performance may vary.*

7. Appendix

Enterprises training on a cloud instance sometimes need to also perform training on-premises. Habana Gaudi based servers can be sourced from OEM partners such as Supermicro.

Figure 5 shows an integrated server configuration with dual socket CPU and eight Gaudi OCP Accelerator Module (OAM) Mezzanine cards. The Gaudi OAM cards are connected all-to-all on the PCB, using seven 100GbE ports of each Gaudi. The all-to-all connectivity allows training across all eight Gaudi processors without requiring an external Ethernet switch. The remaining three ports from each Gaudi are available to scale out the solution over Ethernet ports. The host CPU manages the Gaudi processors through the PCIe ports.

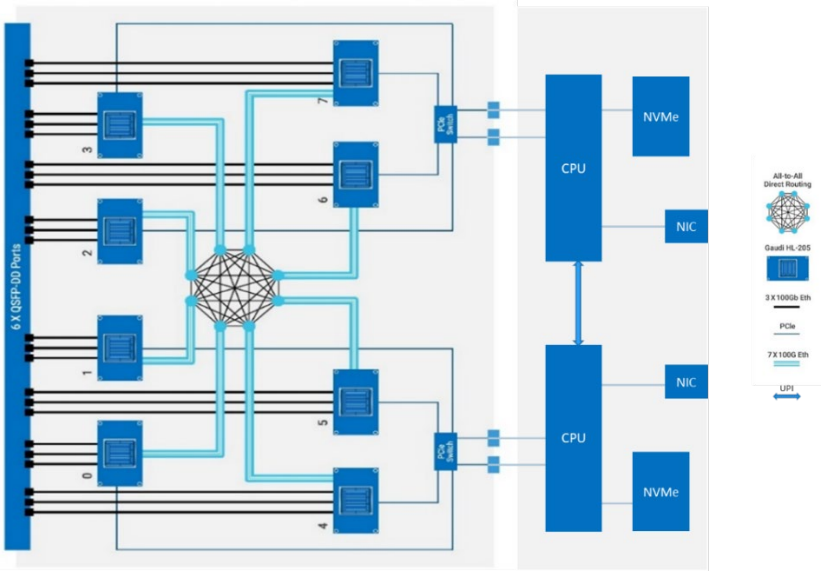

Figure 5. Integrated Server with Gaudi processors, host CPUs and Ethernet Interfaces

Supermicro X12 Gaudi AI Server

The Supermicro X12 Gaudi AI server (SYS-420GH-TNGR), powered by Habana Gaudi AI Processors, pushes the boundaries of deep learning training and can scale up to hundreds of Gaudi processors in one AI cluster. It features eight Gaudi HL-205 mezzanine cards, dual 3rd Gen Intel® Xeon® Scalable processors, two PCIe Gen 4 switches, four hot swappable NVMe/SATA hybrid hard drives, fully redundant power supplies, and 24 x 100GbE RDMA (6 QSFP-DDs) for unprecedented scale-out system bandwidth. This system contains up to 8TB of DDR4-3200MHz memory, unlocking the Gaudi AI processors' full potential. The HL-205 is OCP-OAM (Open Compute Project Accelerator Module) specification compliant. Each card incorporates a Gaudi HL-2000 processor with 32GB HBM2 memory and ten natively integrated ports of 100GbE RoCE v2 RDMA.

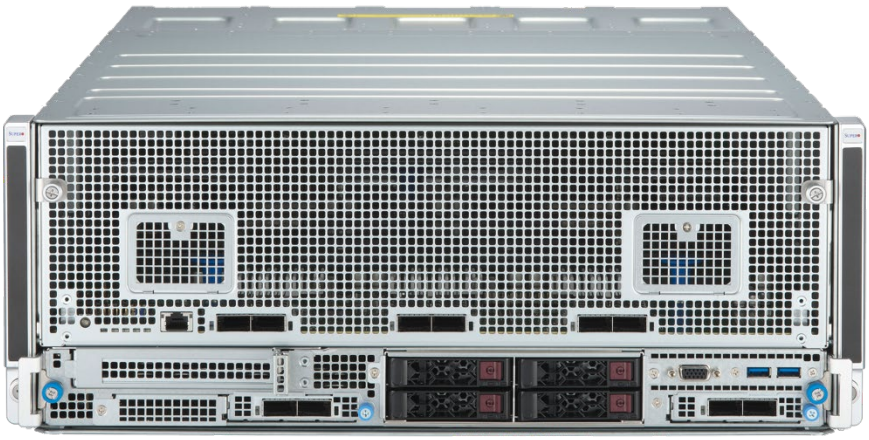

Figure 6. The Supermicro X12 Gaudi AI server

Gaudi HLS-1 Server

HLS-1 is a server designed by Habana Labs, containing eight HL-205 OCP Accelerator Module (OAM) Mezzanine cards and dual PCIe switches. It is a Gaudi-only server, which requires an external host server.

Gaudi HLS1-H Server

HLS-1H is another server designed by Habana Labs, containing four HL-205 OCP Accelerator Module (OAM) Mezzanine cards. The interfaces are 2x16 PCIe Gen4 cables that can be connected to an external host server, and up to 40X100Gb Ethernet links. It is built to enable massive scale-out using off-the-shelf external standard Ethernet switches.

Rack and POD Scale Systems

Customers can easily build training systems at scale using Ethernet switches and a variable number of server nodes. The nodes can be servers composed of integrated CPU, Gaudi and Ethernet interfaces, or combination of Gaudi servers and CPU host servers.

Figure 7 shows a rack-scale configuration with four Gaudi servers (eight Gaudi processors per server) connected to a single Ethernet switch at the top of the rack. This switch can be further connected to other racks in order to form a much larger training pod that can hold hundreds or thousands of Gaudi processors. Each Gaudi server is connected to a CPU host server.

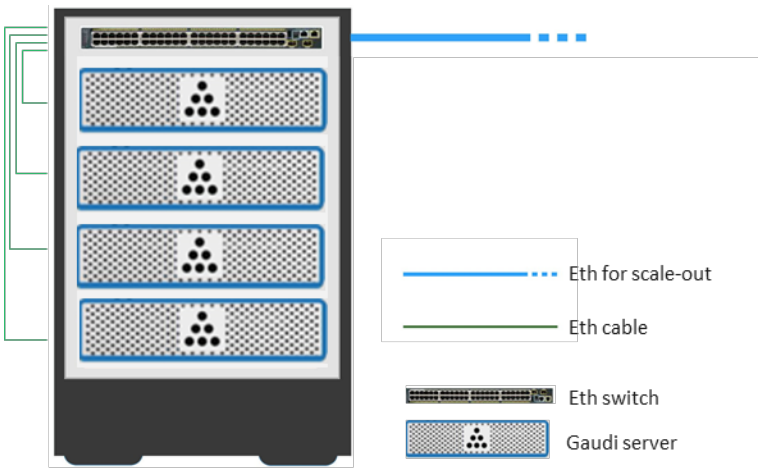

Figure 7: Gaudi Rack-scale Server Configuration Example