Customer Enablement of Habana® Gaudi® Amazon EC2 Instances

December 2020

habana®
An Intel Company

Contents

1. Introduction 1
2. Gaudi Architecture 1
   2.1. Gaudi Processor 1
   2.2. Amazon EC2 Gaudi Instances 2
3. SynapseAI® Software Suite 2
   3.1. Graph Compiler and Runtime 3
   3.2. Habana Communication Libraries 3
   3.3. TPC Programming 4
   3.4. DL Framework Integration 4
   3.5. Embedded Software and Tools 5
4. Habana Developer Platform 5
   4.1. Vault and SynapseAI Container Registry 5
   4.2. Habana GitHub 6
5. Migration to Gaudi 7
6. Appendix 9

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.

2. Gaudi Architecture

2.1. Gaudi Processor

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.

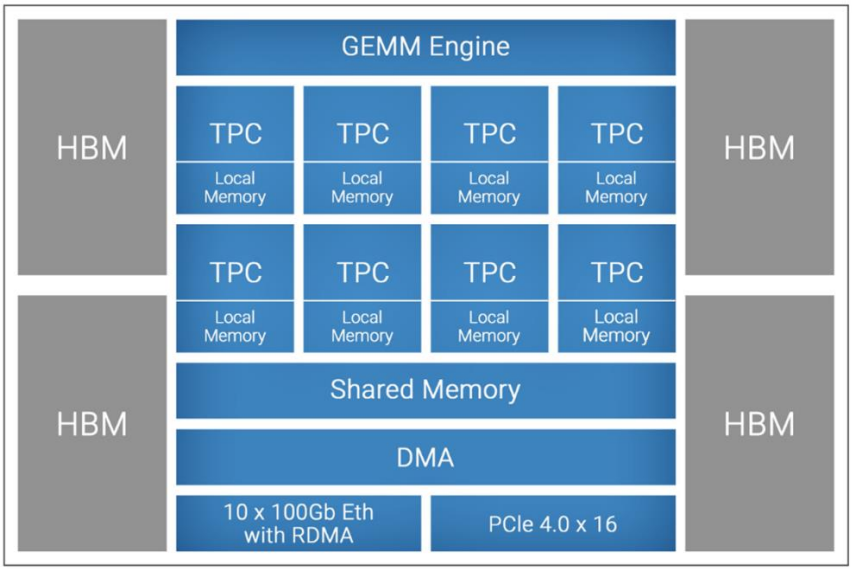

Figure 1: Gaudi Processor High-level Architecture

2.2. Amazon EC2 Gaudi Instances

EC2 instances featuring Habana Gaudi accelerators will feature up to 8 accelerators. These instances will be available for customers as standard EC2 instances via an easy to use and pay-as-you-go usage model. Developers will be able to spin up these instances via AWS ECS and EKS for containerized applications, and also via Amazon SageMaker - a managed service for building, training and deploying machine learning applications

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.

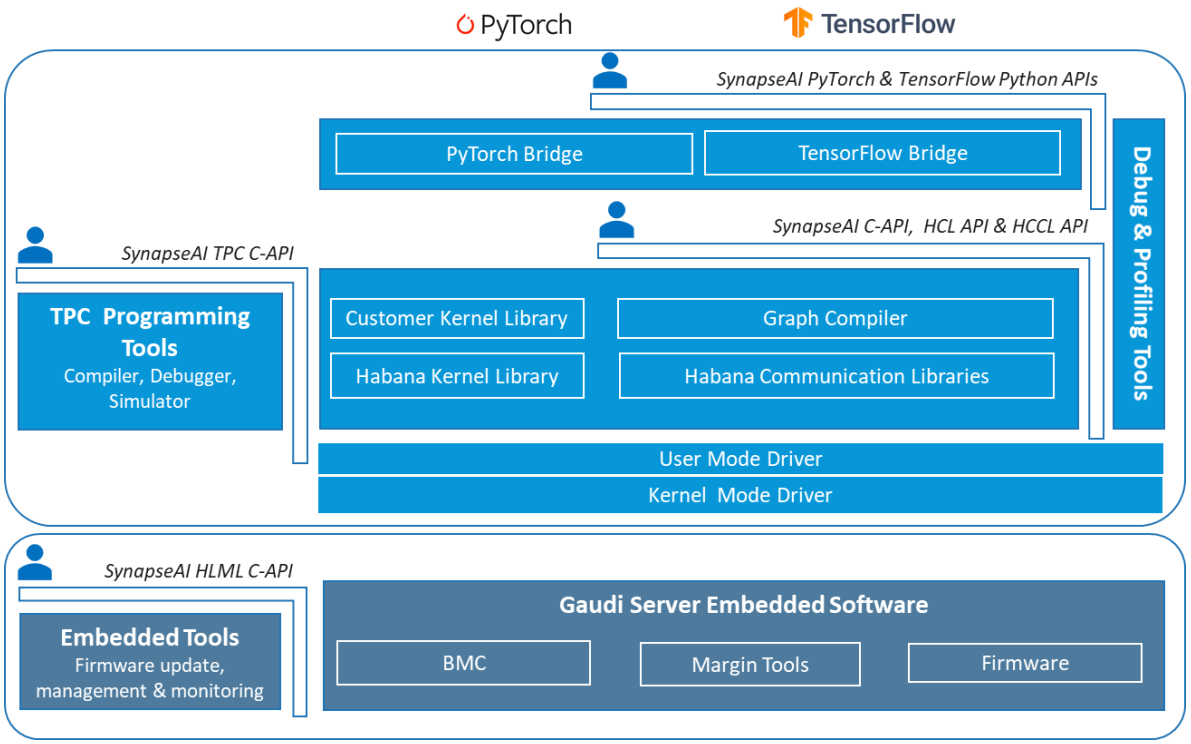

Figure 2: SynapseAI Software Suite

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 (HCL) 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.

The SynapseAI suite also includes Habana Collective Communications Library (HCCL) that is Habana's implementation of standard collective communication routines with NCCL-compatible API. HCL uses

Gaudi integrated NICs for both scale-up and scale-out. HCCL allows users to enable Gaudi integrated NIC for scale-up and host NIC for scale-out.

For efficient scaling across multiple Gaudi-based EC2 Instances, support for AWS Elastic Fabric Adapter will be available

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.

The following section illustrates how Synapse does the work under “the hood” for Tensorflow, 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.

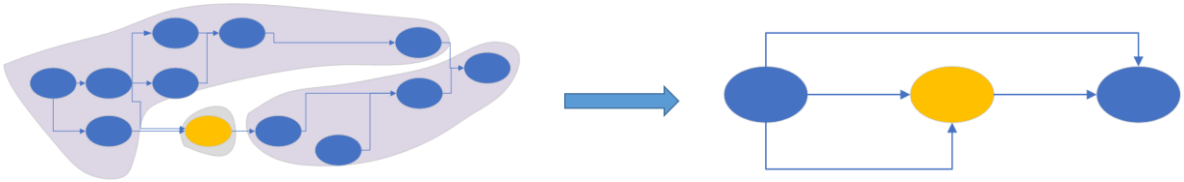

Figure 3. Subgraph selection and encapsulation in TensorFlow framework graph

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 support for monitoring and management that is useful for server developers and IT personnel who manage server deployments. As these are not useful for an EC2 instance users, we have not included them in this particular document.

4. Habana Developer Platform

Developer.habana.ai is the hub for Habana developers from where they will find Gaudi software, optimized models and documentation. It will be fully functional for developer access concurrent with the availability of AWS EC2 Instances based on Gaudi.

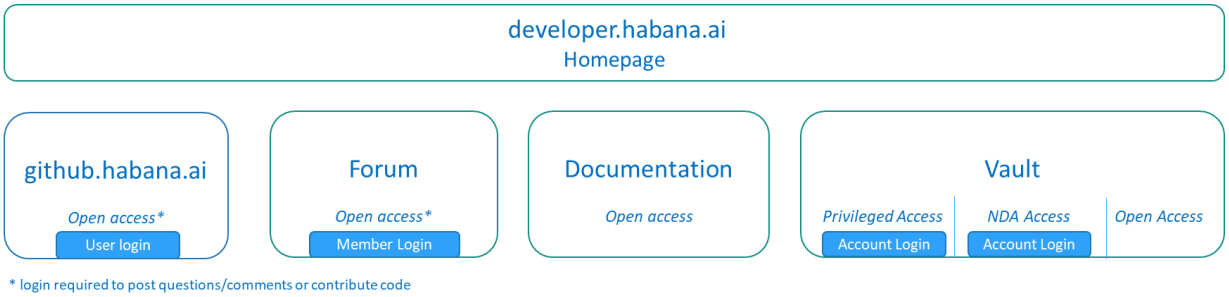

Figure 4: High-level Structure of Habana Developer Platform

Github.habana.ai contains repositories open to the general public. Section 4.2 provides details on the various repositories that will be available. Github.habana.ai will be the primary channel for developer support. 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. Previous versions are not validated and hence not guaranteed to work as expected.

The table below highlights supported versions as of Q4'2020:

Supported Frameworks	TensorFlow (2.2) and PyTorch (1.6)
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 PyTorch distributed (native)

The Vault contains publicly available official releases of SynapseAI TensorFlow and PyTorch Docker container images. For users interested in building their own Docker images, Habana GitHub will have a repository with Dockerfiles and build instructions.

4.2. Habana GitHub

This section provides an overview of select repositories on Habana GitHub, which will be publicly available for developers when AWS EC2 Gaudi Instances become available.

The Setup and Installation repository contains instructions on how to access and install Habana drivers on bare metal. It will also contain Dockerfiles and instructions to build your own Docker images for Synapse AI with TensorFlow or PyTorch. The Getting Started repository contains guidance on setting up the environment with Gaudi firmware and drivers, installing SynapseAI TensorFlow and PyTorch containers, and running with containers.

The SynapseAI software roadmap repository is for communicating the public product roadmap. The GitHub issue tickets will contain brief descriptions outlining the purpose of feature/enhancement and availability timeline. In general, we expect users to have visibility into the plans for at least the upcoming two quarters. The roadmap will be updated on a quarterly basis.

The Reference Models GitHub 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, expected results and tutorials. TensorFlow models will include ResNet50, BERT-Large, ResNext101, SSD-ResNet34 and MaskRCNN. PyTorch models will include BERT-Large and DLRM.

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 roadmap will include updates on new framework operators that we plan to support. Similarly, our TPC kernel library is continually evolving

and growing. We will also provide visibility into the new kernels being planned as part of our regular roadmap updates.

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. 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 will be applied with PyTorch as well.

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 will focus primarily on helping 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.

```
1. import tensorflow as tf
2. from demo.library_loader import load_habana_module
3.
4. tf.compat.v1.disable_eager_execution()
5. load_habana_module()
6.
7. (x_train, y_train), (x_test, y_test) = tf.keras.datasets.mnist.load_data()
8. x_train, x_test = x_train / 255.0, x_test / 255.0
9.
10.model = tf.keras.models.Sequential([
11.    tf.keras.layers.Flatten(input_shape=(28, 28)),
12.    tf.keras.layers.Dense(10),
13.])
14.loss = tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True)
15.optimizer = tf.keras.optimizers.SGD(learning_rate=0.01)
16.
17.model.compile(optimizer=optimizer, loss=loss, metrics=['accuracy'])
18.
19.model.fit(x_train, y_train, epochs=5, batch_size=128)
20.model.evaluate(x_test, y_test)
```

The changes to enable Habana Gaudi are highlighted in **bold:**

- On line 2, we import ‘load_habana_module’ needed to enable Gaudi
- On line 4, we disable eager execution; this is needed to enable SynapseAI optimization passes in TensorFlow
- On line 5, we now call the ‘load_habana_module()’ function to enable Gaudi (registered as ‘HPU’ device in TF).

Upon successful execution of ‘`python3 example.py`’ the following lines are printed as part of output:

```
Train on 60000 samples
Epoch 1/5
60000/60000 [==============================] - 1s 16us/sample - loss: 2.2734 - accuracy: 0.2589
Epoch 2/5
60000/60000 [==============================] - 1s 16us/sample - loss: 2.1497 - accuracy: 0.5032
Epoch 3/5
60000/60000 [==============================] - 1s 15us/sample - loss: 2.0143 - accuracy: 0.6493
Epoch 4/5
60000/60000 [==============================] - 1s 16us/sample - loss: 1.9218 - accuracy: 0.6960
Epoch 5/5
60000/60000 [==============================] - 1s 18us/sample - loss: 1.8735 - accuracy: 0.7056
Train on 60000 samples
```

There will be similar examples for training on multiple Gaudi devices in a single server (scale-up), as well as multi-node training (scale-out.)

The migration guide will include practical steps to ensure accuracy such as removing randomness from the model and comparing side by side with GPUs for single and multiple iterations. Performance analysis and improvement topics include operator coverage, identifying the operators not supported on Gaudi (if any) and adding custom operators to improve performance. Other topics include leveraging mixed precision by enabling BF16 as numeric data type, and recipes for typical vision and language models. There will be practical tips on debug and profiling. This includes using framework tools and APIs (TF debug, TensorBoard) as well as SynapseAI profiler.

We will have a similar approach to enable developers familiar with programming on other platforms to be productive on Gaudi with TPC programming for custom kernels. We will publish a migration guide practical tips and guidelines, in addition to the TPC programming user guide, training videos, sample code and so on.

6. Appendix

Scaling-out an AI workload has never been easier. Gaudi leverages a superior, open-standard networking technology for scale-out. Each Gaudi chip implements 10 ports of standard 100Gbit Ethernet. Integrating networking directly into the AI processor chip creates a nimble system without bandwidth bottlenecks. By combining multiple Gaudi chips with Ethernet switching, limitless possibilities are available for distributing training across 8, 16, 32, 64, 128, 1K, 2K, 8K and more Gaudi chips.

For customers interested in on-premise installation, the following Gaudi server configurations will be available through Habana's OxM partners.

Gaudi Server with CPU

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.

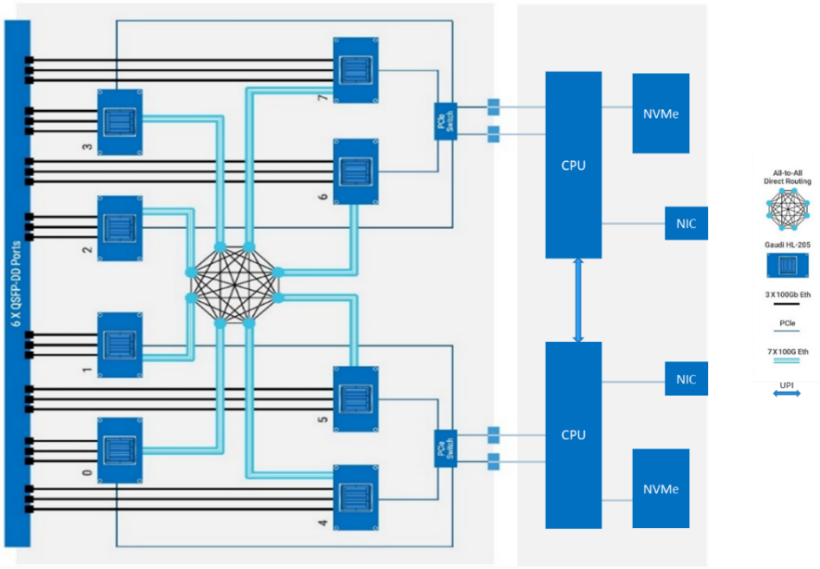

Figure 5. Integrated Server with Gaudi processors, host CPUs and Ethernet Interfaces

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 6 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.

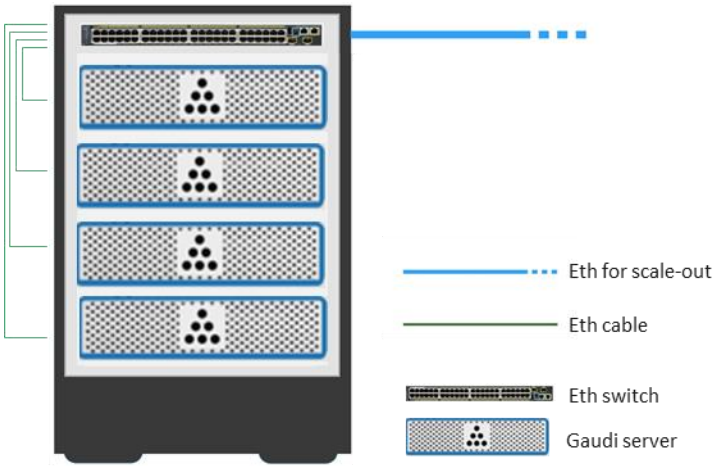

Figure 6: Gaudi Rack-scale Server Configuration Example