



GAUDI[®]2

HABANA[®] GAUDI[®]2

WHITE PAPER

JUNE 2022





Table of Contents

I.	INTRODUCTION	3
II.	GAUDI®2 CHIP ARCHITECTURE	6
III.	SYNAPSEAI® SOFTWARE SUITE	11
	GRAPH COMPILER AND RUNTIME	12
	HABANA COMMUNICATION LIBRARIES	12
	TPC PROGRAMMING	12
	DL FRAMEWORK INTEGRATION	12
	SYSTEM MANAGEMENT AND MONITORING	14
	ORCHESTRATION	14
IV.	HABANA DEVELOPER SITE	15
V.	MODEL MIGRATION	17
VI.	ECOSYSTEM PARTNERSHIPS	19
	HUGGING FACE TRANSFORMERS WITH GAUDI.....	20
	PYTORCH LIGHTNING WITH GAUDI.....	22
	MLOPS WITH CNVRG.IO	23
VII.	RACK LEVEL INTEGRATION	24
	DDN A3I REFERENCE ARCHITECTURE FOR GAUDI AI SERVERS	24
VIII.	THE GAUDI®2 OAM CARD	29
IX.	GAUDI®2 HLBA-225 BASEBOARD	30
	BLOCK DIAGRAM AND MAIN COMPONENTS	30
X.	HLS-GAUDI®2 SERVER.....	33



I. INTRODUCTION

Gaudi®2 is Habana's second-generation deep learning accelerator supporting Training and Inference.

Building on the architecture of Gaudi®, which launched first on the AWS EC2 cloud in the DL1 instance, and on-premises via the Supermicro X12 Gaudi Training Server, Gaudi2 brings a new level of performance and efficiency to deep learning in the datacenter and cloud.

Gaudi2 is supported by the SynapseAI® software suite, which is integrated with TensorFlow and PyTorch frameworks. We offer reference models, tools and how-to-guides on Habana's GitHub and Developer Site.

The main benefits that current customers of our first-generation Gaudi see are the price-performance advantages relative to GPU solutions for popular vision and language models. These enable customers to train more and pay less, and this way, accelerate time-to-market with their model training.

With Gaudi2, we are pleased to extend these benefits to not just better price-performance, but also performance leadership vs. the leading shipping 7nm GPU today, namely the Nvidia A100.



Before we address the architecture details, here are key benchmarks for Gaudi2 at time of publication of this white paper and which reflect the SynapseAI® Software Suite Release 1.5 effective June 16, 2022

FIGURE 1. RESNET-50 TRAINING THROUGHPUT

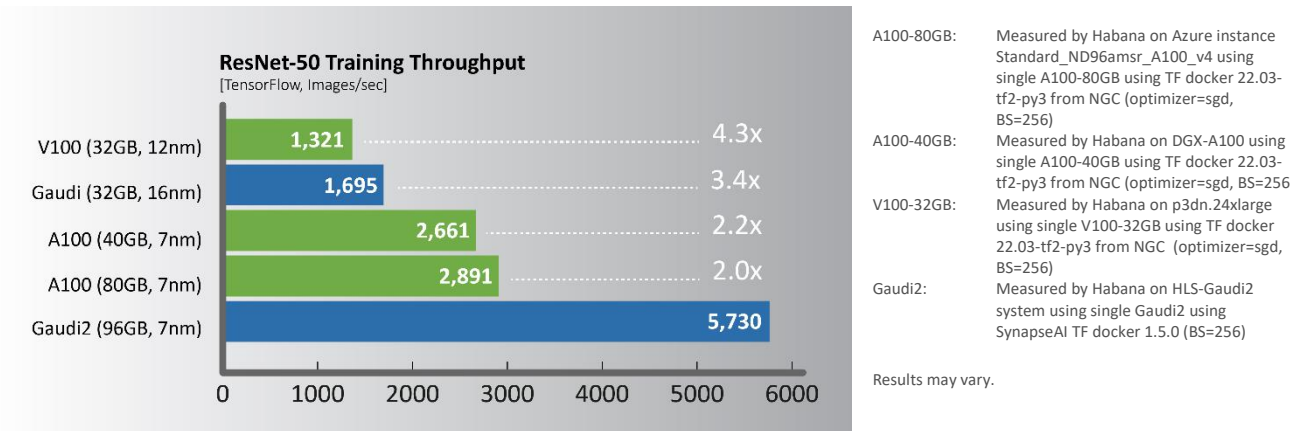


FIGURE 2. BERT PHASE-1 TRAINING THROUGHPUT

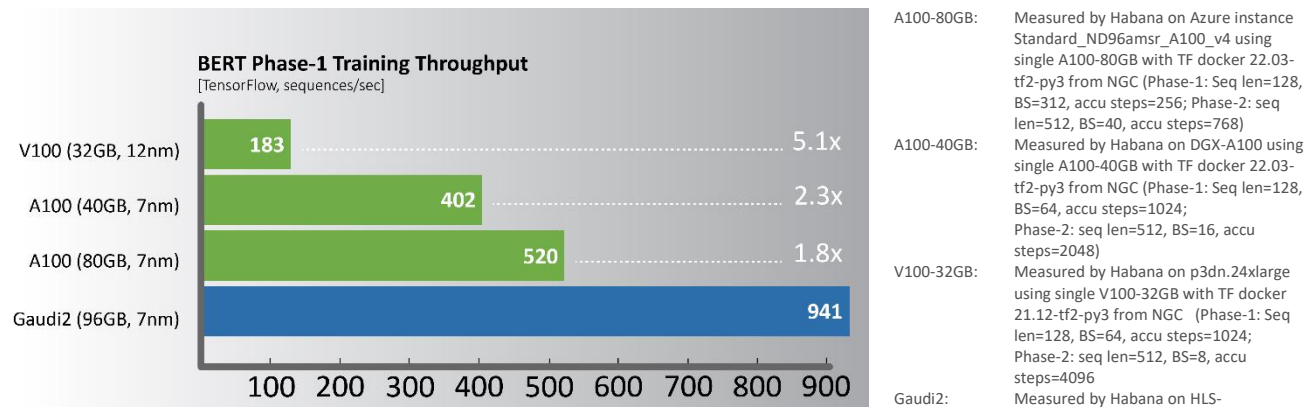
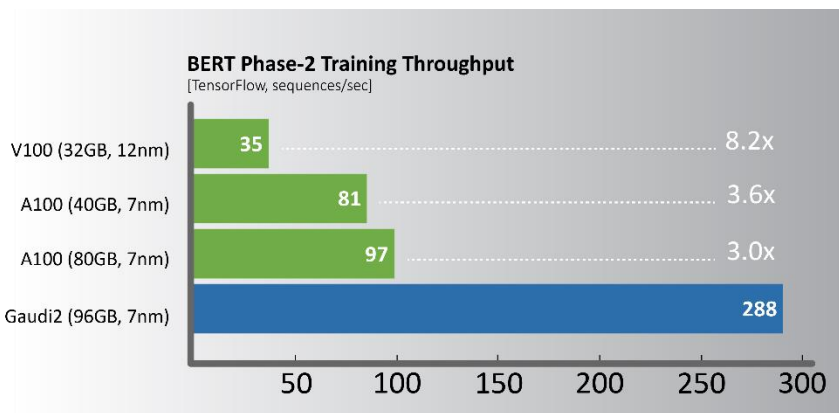


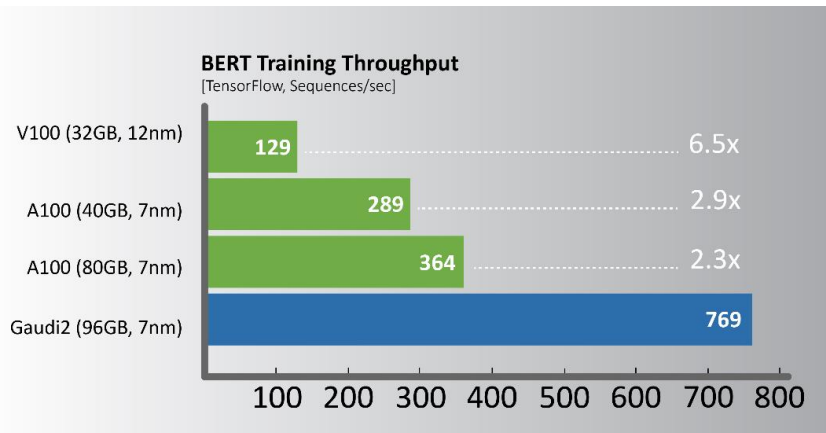
FIGURE 3: BERT PHASE-2 TRAINING THROUGHPUT





And combining Phase-1 and Phase-2 of the BERT-Large model:

FIGURE 4. BERT TRAINING THROUGHPUT



A100-80GB: Measured by Habana on Azure instance Standard_ND96amsr_A100_v4 using single A100-80GB with TF docker 22.03-tf2-py3 from NGC (Phase-1: Seq len=128, BS=312, accu steps=256; Phase-2: seq len=512, BS=40, accu steps=768)

A100-40GB: Measured by Habana on DGX-A100 using single A100-40GB with TF docker 22.03-tf2-py3 from NGC (Phase-1: Seq len=128, BS=64, accu steps=1024; Phase-2: seq len=512, BS=16, accu steps=2048)

V100-32GB: Measured by Habana on p3dn.24xlarge using single V100-32GB with TF docker 21.12-tf2-py3 from NGC (Phase-1: Seq len=128, BS=64, accu steps=1024; Phase-2: seq len=512, BS=8, accu steps=4096)

Gaudi2: Measured by Habana on HLS-Gaudi2 system using single Gaudi2 with SynapseAI TF docker 1.5.0 (Phase-1: Seq len=128, BS=64, accu steps=1024; Phase-2: seq len=512, BS=16, accu steps=2048)

Results may vary.

Habana reported these results on models ported from Gaudi to Gaudi2 and based on the SynapseAI® Software Suite release 1.5. While our OEM partners are working on building servers for general consumption, Habana's engineers are working these days on porting and developing additional deep-learning models, with a cadence of 6-8 weeks for software releases. You can track our progress through our public Github and developer.habana.ai site.

Now let's dive in to the Gaudi2 processor, hardware, and software.



II. GAUDI®2 CHIP ARCHITECTURE

Gaudi2 architecture includes 3 main subsystems:
Compute, memory, and networking.

The compute architecture is heterogeneous and includes two compute engines – a Matrix Multiplication Engine (MME) and a fully programmable Tensor Processor Core (TPC) cluster. The MME is responsible for doing all operations which can be lowered to Matrix Multiplication (fully connected layers, convolutions, batched-GEMM) while the TPC, a VLIW SIMD processor tailor-made for deep learning operations, is used to accelerate everything else. Besides MME & TPC, Gaudi2 is also instantiating several DMAs which are coupled with a transpose engine for efficient, on the fly, tensor shape transformations, in addition to the ability to read & write non-contiguous multi-dimensional tensors from and to the Gaudi2 memory subsystem.

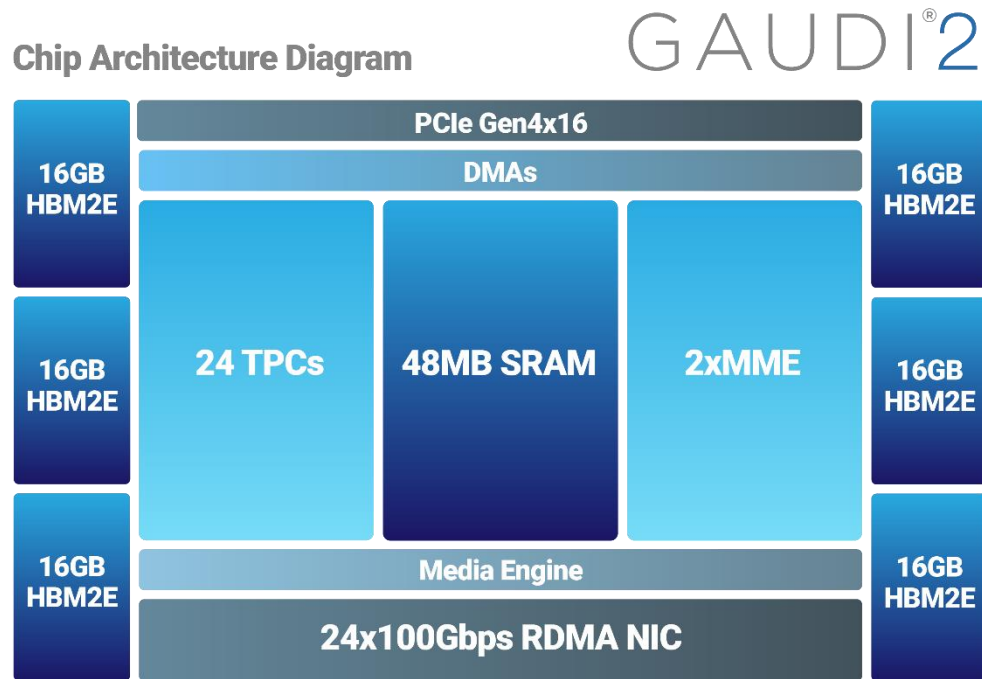
The Gaudi2 Processor offers 2.4 Terabits of networking bandwidth with the native integration on-chip of 24 x 100 Gbps RoCE V2 RDMA NICs, which enable inter-Gaudi communication via direct routing or via standard Ethernet switching.

The Gaudi2 Memory subsystem includes 96 GB of HBM2E memories delivering 2.45 TB/sec bandwidth, in addition to a 48 MB of local SRAM with sufficient bandwidth to allow MME, TPC, DMAs and RDMA NICs to operate in parallel.

Specifically for Vision applications, Gaudi2 has integrated media decoders which operate independently and can handle the entire pre-processing pipe in all popular formats – HEVC, H.264, VP9 & JPEG as well as post-decode image transformations needed to prepare the data for the AI pipeline.

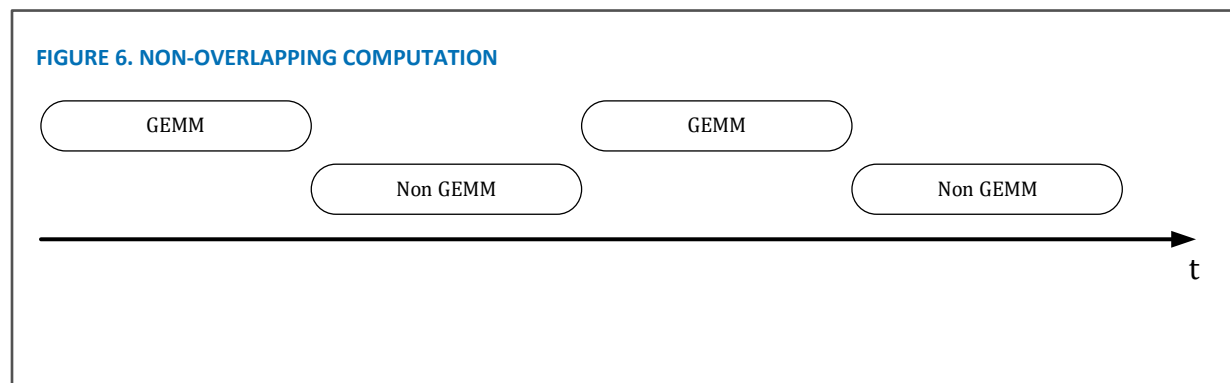
Gaudi2 supports all popular data types required for deep learning: FP32, TF32, BF16, FP16 & FP8 (both E4M3 and E5M2). In the MME, all data types are accumulated into an FP32 accumulator.

FIGURE 5. GAUDI2 ARCHITECTURE DIAGRAM

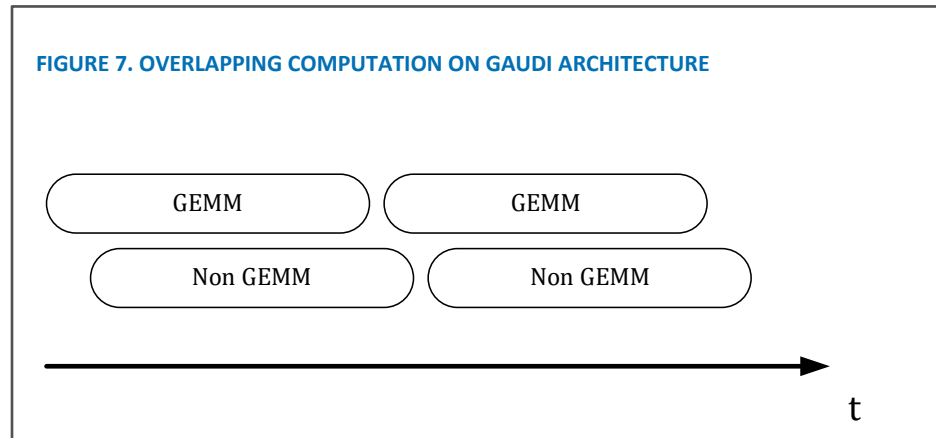


Gaudi2 architecture is a heterogenous architecture with primarily two types of compute engines – MME & TPC. Gaudi 2 was architected to enable parallel operation of MME and TPC, such that their compute time is overlapped to accelerate the execution of the deep learning topology dramatically.

The below diagram shows what is usually observed on GPUs where the GEMM compute and the general-purpose cores execution time is not overlapping:

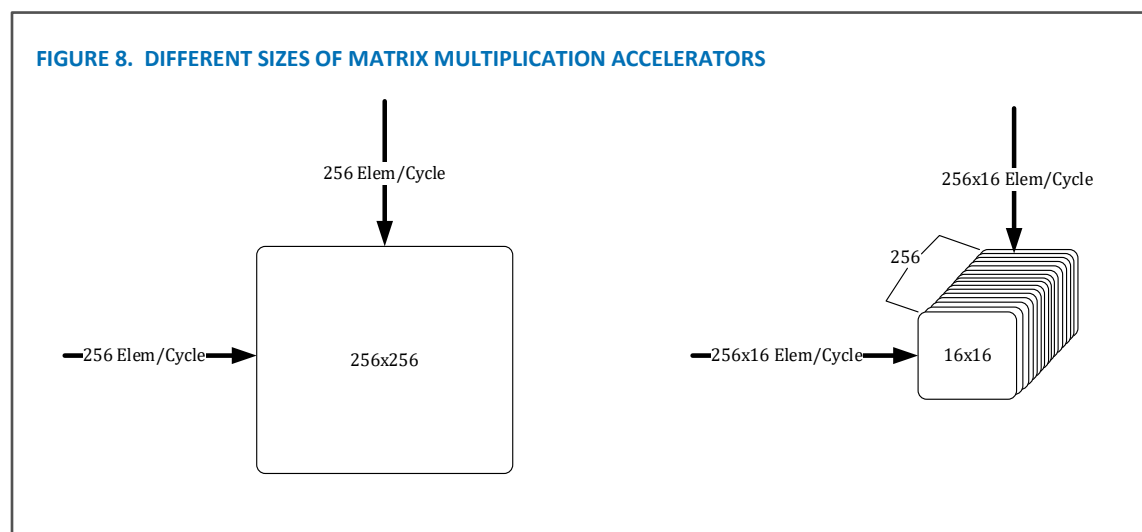


On Gaudi and Gaudi2 architectures, the MME and TPC compute time overlaps. The GEMM and non-GEMM operations are mostly overlapped, dramatically accelerating the workload.



Another big difference between GPU and Gaudi architecture is the size of the matrix multiplication accelerator. This fact, which may seem minor, has a big effect on overall ability to utilize those accelerators, specifically when matrix sizes become smaller.

The below diagram compares a 256x256 matrix accelerator (on the left) to 256 small 16x16 matrix accelerators on the right (the depth dimension was removed to simplify the explanation). From compute perspective, both are equivalent, but from bandwidth perspective, although the left one requires 512 input elements per cycle to utilize the compute, the right side requires 8K input elements per cycle to utilize the compute. A 16x difference on the read bandwidth requirements towards the first level memory.

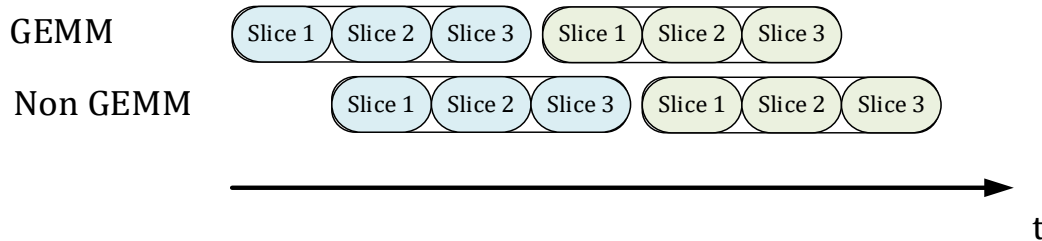


GPUs, which implement the right side, compensate for the above phenomena, by mandating a large reuse factor from the hierarchical caching memory sub system, such that overall, to utilize their multipliers, they require a very big matrix multiplication problem to solve. Gaudi2 (and some other dedicated tensor processors), which implement the left side approach, can utilize their multipliers easily while leaving a lot of free bandwidth & capacity from their flat memory subsystem for other tasks, besides matrix multiplications.

The above difference between a big matrix multiplication accelerator to a lot of small ones can be observed in the utilization each architecture can achieve with respect to matrix size. To utilize 80% of modern GPUs' many small matrix multiplication accelerators, a GEMM dimension of $m=n=k \sim 3K$ is required. On Gaudi2, $m=n=k=1K$ is sufficient to utilize 100% of the multipliers. And if activations are pipelined via 48MB SRAM (which is usually the case), $m=n=k=512$ is sufficient to utilize MME by 100%. In other words, Gaudi2 requires between $\sim 25x \sim 200x$ less MACs in a GEMM operation to be utilized 100% compared to modern GPUs which are utilized 80%. Paradoxically, creating a relatively big matrix multiplication accelerator allows hardware to be efficiently utilized on smaller tensors compared to the alternative.

Such high utilization on small tensors significantly eases MME & TPC computation overlapping, as to allow such tight overlapping, an operation needs to be sliced as described below, which creates smaller tensors for MME & TPC to operate upon:

FIGURE 9. SLICED COMPUTATION OVERLAPPING





Gaudi2 integrates Habana's fourth generation Tensor Processor Core. The TPC is a general purpose VLIW processor which is 256B SIMD wide and supports FP32, BF16, FP16 & FP8 (Both E4M3 and E5M2), in addition to INT32, INT16 & INT8 data types. As opposed to common DSPs, which require a DMA to fetch in and out the operands to a local SRAM, the TPC, using advanced micro-architectural techniques, exposes a DMA-free programming model which significantly eases SW development. In addition, the same advanced microarchitecture allows bubble-free execution between kernels which effectively makes TPC 100% utilized on tensor processing, even for very short kernels, regardless of the location of its inputs/outputs (SRAM or HBM). Just like MME, TPC is also very efficient in working on small tensors.

As deep learning training is usually solved on multiple devices, Gaudi2 Network Interface controllers (NICs) are an essential component in the overall Habana second-generation training solution. Gaudi's NIC is customized to fit a distribution of a DNN graph between the chips in the network (AKA scale-out). The NIC provides the compute engine with a remote direct memory access (RDMA) featuring high bandwidth and low latency over reliable connection without any software intervention. To fit common cloud infrastructure, NIC ports use Ethernet connectivity with aggregated bandwidth of 2.4Tb/s, supporting multiple port configurations. The NIC implements the RoCE v2 specification, benefiting from the commonly used Ethernet infrastructure and the reliable and low latency RDMA of the InfiniBand protocol, while

extending the RoCE scalability with flexible time-based congestion control algorithm, enabling linear scalability over thousands of Gaudi system.

DNN topologies tend to use collective operations extensively and posting collective operations on multiple ports usually requires high CPU horsepower. To reduce the CPU utilization a scalable collective offload was introduced on Gaudi2 which helps Gaudi's 2 message rate to be more than an order of magnitude better than competition. Gaudi's NICs are also aligned with all other engines in chip and can access both local and remote memory in tensor semantics.

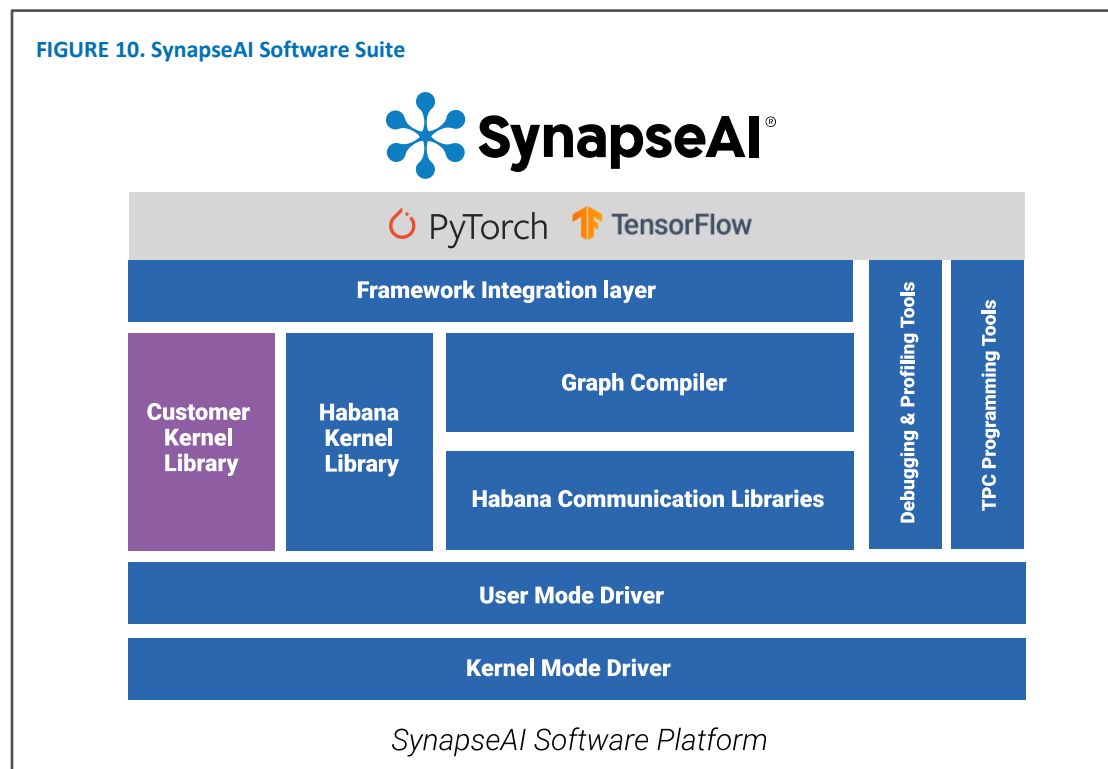
To summarize, Gaudi heterogeneous architecture is unique in the sense that it is highly efficient on small tensors' operations, which is an enabler for overlapping the computation & networking communication between the heterogeneous agents, in addition to freeing up significant memory capacity and bandwidth requirements from its memory subsystem.

III. SYNAPSEAI® SOFTWARE SUITE

Designed to facilitate high-performance deep learning (DL) training on Habana's AI processors, the SynapseAI® Software Suite enables efficient mapping of neural network topologies onto the Gaudi hardware family.

The software suite includes Habana's graph compiler and runtime, TPC kernel library, firmware and drivers, and developer tools such as the TPC programming tool kit for custom kernel development and SynapseAI Profiler. SynapseAI is integrated with the popular frameworks, TensorFlow and PyTorch, and performance-optimized for Habana's Gaudi and Gaudi2 AI processors. Figure 10 shows a pictorial view of the SynapseAI software suite.

FIGURE 10. SynapseAI Software Suite





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 performance-optimized operations (for example, elementwise, non-linear, non-GEMM operators). Given the heterogeneous 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, 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.

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.

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, acceleration for special functions, random number generation and multiple data types.

DL Framework Integration

Habana SynapseAI integrates PyTorch and TensorFlow, two of the most popular frameworks used by data scientists and AI developers. 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 11 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.

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.

SynapseAI supports distributed training with TensorFlow using Horovod and `tf.distribute` API with `HPUStrategy`. Mixed precision execution is available via the `tf.keras.mixed_precision` API or using Habana’s automated mixed precision conversion. These enable you to run mixed precision training without extensive modifications to existing FP32 model scripts. More details are available in the TensorFlow section on docs.habana.ai.

The SynapseAI PyTorch bridge interfaces between the framework and SynapseAI software stack to train PyTorch-based deep learning models on Gaudi. We support two modes of execution: (1) Eager mode, which performs operator-by-operator execution as defined in standard PyTorch eager mode scripts, and (2) Lazy mode, which performs deferred execution of graphs comprising a collection of operators. Lazy mode provides user experience like Eager mode, while enabling high

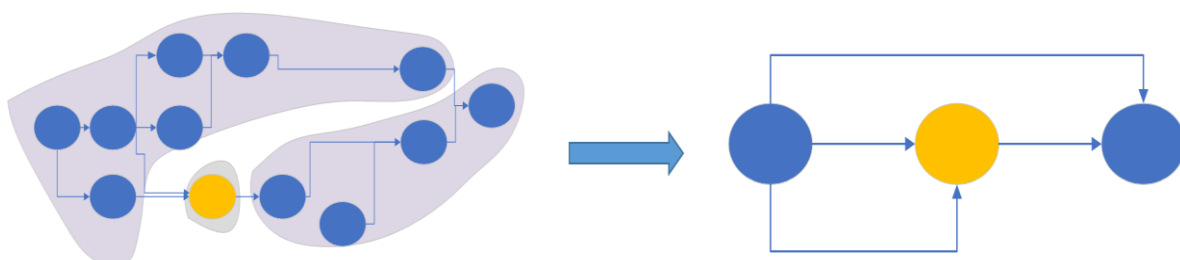
performance on Gaudi. By default, we enable lazy mode execution. Instead of executing one operator at a time, the SynapseAI bridge internally accumulates these in a graph.

The execution of the accumulated operators in the graph is triggered in a “lazy” manner, only when a tensor value is required by the user. This allows the bridge to construct a graph, which provides the SynapseAI graph compiler the opportunity to optimize the device execution for the operators.

Mixed precision execution is available via the Habana Mixed Precision (HMP) package. The HMP package automatically modifies the Python operators to add the appropriate cast operations, and this enables you to run mixed precision training without extensive modifications to existing FP32 model scripts. SynapseAI PyTorch bridge supports distributed training using `torch.distributed` and `torch.nn.parallel.DistributedDataParallel` APIs for both data and model parallelism. Distributed communication is enabled using HCCL backend. For more details, check out the PyTorch section on docs.habana.ai.

SynapseAI is also integrated with TensorBoard to enable debugging and profiling of your TensorFlow or PyTorch models. Users interested in low-level focused profiling can refer to the SynapseAI Profiler User Guide on docs.habana.ai.

FIGURE 11. SUBGRAPH SELECTION AND ENCAPSULATION IN TENSORFLOW FRAMEWORK GRAPH





System Management and Monitoring

The SynapseAI software suite includes support for monitoring and management that is useful for server developers and IT personnel who manage server deployments. The Habana Labs Management Library (HLML) is a C-based programmatic interface for monitoring and managing various states within Habana AI processors. The Habana System Management Interface (hl-smi) is a command line utility, based on top of HLML, intended to aid in the management and monitoring of Habana AI processors. The Habana Labs Qualification (hl_qual) tools package provides the required tools to validate and qualify the usage and integration of Gaudi hardware platforms in your server design.

Orchestration

Kubernetes is an orchestration system that automates the processes of running containers in production clusters. It eliminates the infrastructure complexity associated with deploying, scaling, and managing containerized applications. Kubernetes based container orchestration is commonly used for deploying AI workloads, and SynapseAI provides the necessary components to enable Gaudi in a Kubernetes cluster. The Habana Device Plugin enables the registration of the Habana device in a container cluster for compute workload. With the appropriate hardware and this plugin deployed in your Kubernetes cluster, you will be able to run jobs that require a Habana device. Habana uses the standard MPI Operator from Kubeflow that enables the running of MPI all-reduce style workloads leveraging Gaudi processors in a Kubernetes cluster. This enables you to run distributed training on Gaudi with the Kubernetes job distribution model. To enable monitoring of cluster health, SynapseAI also includes support for Prometheus Metric Exporter for Kubernetes. It is a Daemonset that enables the collection of device metrics in a container cluster for compute workload. SynapseAI supports multiple flavors of Kubernetes orchestration, including vanilla open-source Kubernetes, Amazon EKS, RedHat OpenShift and VMware Tanzu.



IV. HABANA DEVELOPER SITE

At Habana, designing and developing the hardware for high-performance and efficient DL processors accounts for a relatively small portion of Habana's effort; the majority is dedicated to leveraging that hardware with the right software, tools and support you need to make your workloads and models run efficiently, with accuracy and speed.

In addition to the SynapseAI software suite that is designed for performance and usability, we have also published a wealth of information and resources to make it easy for you to get started with training on Gaudi processors. The [Habana Developer Site](#), is the hub for Habana developers from where you will find the content, guidance, tools, and support needed to build easily and flexibly new or migrate existing AI models and optimize their performance on our AI processors.

The Resources section contains a collection of documents, short videos and hands-on Jupyter notebook tutorials to help you get started and

running models on Gaudi. And for IT and Systems Administrators building Gaudi-based systems on premise, we provide guidance on set-up and management for Gaudi servers and computing infrastructure.

[Habana GitHub](#) contains repositories open to the general public, which include setup and install instructions for Habana binaries and docker creation, Jupyter notebook-based tutorials, reference models, custom TPC kernel example, and more.

Our [Model-References repository](#) contains 30+ popular TensorFlow and PyTorch models that have been ported to Gaudi, and the [Model Performance](#) page provides the latest performance results for these models. The Habana Developer Site also has a searchable [Catalog](#) of SynapseAI container images, TensorFlow and PyTorch reference models. For more information on future model support, please refer to our [SynapseAI model roadmap page](#). Each model is supported with model scripts and instructions on how to run these models on Gaudi. We are committed to expanding our model coverage continuously and providing a wide variety of examples for users.



[Habana's Documentation](#) page hosts detailed documentation for SynapseAI User Guides, TPC User Guides, API Reference Guides and Migration Guides. It is web-based and searchable with content based on the latest software release and with archive of the prior software release documentation. We also provide developers with "Documentation Update," which summarizes major updates from the previous software release.

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 frameworks containers are delivered with all necessary dependencies including the complete SynapseAI software.

SynapseAI is integrated and validated with recent versions of TensorFlow and PyTorch. Support for additional framework operators, features and APIs will continue to evolve over time. Please refer to the [Support Matrix](#) on the Habana Documentation.

The table below highlights supported versions as of April 2022:

TABLE 1. SYNAPSEAI SUPPORT MATRIX

Supported Frameworks	TensorFlow2 and PyTorch
Operating Systems	Ubuntu 18.04 and 20.04, AWS Linux2, RHEL8
Container Runtimes	Docker
Distributed Training Schemes	TensorFlow: Horovod and tf.distribute PyTorch: Distributed data parallel (DDP)
Orchestration	Kubernetes

The [Habana Community Forum](#) is a dynamic resource for the developer community to access answers to their questions when implementing or managing Gaudi-based systems, and to share their own insights and perspectives with others who are working with Habana and Gaudi. We invite you to join the Forum and help build a robust and vital community of AI thought-leaders and builders who seek to leverage the unprecedented benefits of Habana's AI processors.



V. MODEL MIGRATION

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. More information on migrating models to Gaudi is available in the Migration Guide on docs.habana.ai.

The SynapseAI TensorFlow and PyTorch user guides provide an overview of SynapseAI integration with respective frameworks, APIs and operators that are supported, how to enable mixed precision and distributed training, and more. The migration guides help users develop a better understanding of how to port their existing models to Gaudi and provide practical tips to assist in their effort.



Below we show the minimum set of changes required to port 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)
```

The minimal changes to enable training on the Habana Gaudi device are highlighted in **bold**. All you need is to import `load_habana_module` package and then invoke the `load_habana_module()` function to enable training on Gaudi. With this change, the Gaudi device, which is referred to as **HPU** in the framework, is now registered in TensorFlow and prioritized for execution 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 Gaudi, it runs on the CPU. For more details on porting your TensorFlow model to Gaudi processors, check out the [TensorFlow Migration Guide](#).

A similar approach applies to migrating your PyTorch models as well. More information on migrating PyTorch models to Gaudi processors is available in the [PyTorch Migration Guide](#) on docs.habana.ai.



VI. ECOSYSTEM PARTNERSHIPS

The AI software ecosystem is rapidly expanding with research breakthroughs being quickly integrated into popular software packages in a scalable and hardware agnostic fashion.

Data scientists and AI developers are adopting these software solutions to help them focus more on the data science and research, and less on managing the complexities of underlying software engineering.

At Habana, our aim is to meet the developers where they are. We have been busy collaborating with AI software ecosystem partners to enable a seamless user experience with Habana AI processors.



Hugging Face Transformers with Gaudi

Powered by deep learning, transformer models deliver state-of-the-art performance on a wide range of machine learning tasks, such as natural language processing, computer vision, speech, and more. With 60,000+ stars on Github, 30,000+ models, and millions of monthly visits, Hugging Face is one of the fastest-growing projects in open source software history, and the go-to place for the machine learning community. With the integration of Habana's SynapseAI software suite with the Hugging Face Optimum open-source library, data scientists and machine learning engineers can now accelerate their Transformer training jobs on Habana AI processors with just a few lines of code and enjoy greater productivity as well as lower training cost.

There are two main classes one needs to know: (1) **GaudiTrainer** class that takes care of compiling (lazy or eager mode) and distributing the model to run on HPUs, and of performing training and evaluation, and (2) **GaudiConfig** class to configure Habana Mixed Precision and decide whether optimized operators and optimizers should be used. The GaudiTrainer is very similar to the Transformers Trainer and adapting a script using the Trainer to make it work with Gaudi will mostly consist in simply swapping the Trainer class for the GaudiTrainer one. The example in the picture above shows how simple it is to get started with training Transformer models on Gaudi. Several popular reference models are available on the [HuggingFace Habana](#) page, including, bert-base, bert-large, roberta-base, roberta-large, distilbert-base, albert-large and albert-xxlarge.



```
from optimum.habana import GaudiConfig, GaudiTrainer, GaudiTrainingArguments

# A lot of the same code as the original script here

...

# Loading the GaudiConfig needed by the GaudiTrainer to fine-tune the model on HPUs
gaudi_config = GaudiConfig.from_pretrained(

    training_args.gaudi_config_name,

    cache_dir=model_args.cache_dir,

    revision=model_args.model_revision,

    use_auth_token=True if model_args.use_auth_token else None,

)

# Initialize our Trainer
trainer = GaudiTrainer(

    model=model,

    gaudi_config=gaudi_config,

    # The training arguments differ a bit from the original ones, that is why we use
    GaudiTrainingArguments

    args=training_args,

    train_dataset=train_dataset if training_args.do_train else None,

    eval_dataset=eval_dataset if training_args.do_eval else None,

    compute_metrics=compute_metrics,

    tokenizer=tokenizer,

    data_collator=data_collator,

)
```



PyTorch Lightning with Gaudi

PyTorch Lightning is a lightweight framework built on PyTorch and provides APIs that abstract the boilerplate code that PyTorch users need to train models. Lightning adoption has grown quickly in the last two years, with over 600 contributors, 15K GitHub stars, and 2 million monthly downloads, with 10x yoy growth. Habana has been collaborating with Grid.ai to make it easier and quicker for developers to train on Gaudi® processors with PyTorch Lightning without any code changes. Grid.ai and PyTorch Lightning make coding neural networks simple. Habana Gaudi makes it cost efficient to train those networks. The integration of Habana's SynapseAI® software suite with PyTorch Lightning brings the best of both worlds together, enabling greater developer productivity while lowering the cost of model training. [PyTorch Lightning 1.6](#) that was recently released now supports Habana Gaudi. Developers now have the flexibility to choose Gaudi's AI computational power with Lightning to benefit from Gaudi advantages with speed and ease.

Below you will see how easy it is to get started with training on Gaudi using PyTorch Lightning:

```
import pytorch_lightning as pl

from pytorch_lightning.plugins import HPUPrecisionPlugin

trainer = pl.Trainer(accelerator="hpu", devices=8, precision=16)
```

All you need is to provide `accelerator="hpu"` parameter to the Trainer class, and select the number of Gaudi processors by setting the `devices` parameter. For mixed precision training, import the `HPUPrecisionPlugin` and set `"precision=16"`.



MLOps with cnvrg.io

[cnvrg.io](#) is a machine learning platform built by data scientists, for data scientists. cnvrg.io is transforming the way enterprises manage, scale and accelerate AI and data science development from research to production and offers unrivaled flexibility to run on-premise, cloud or both. Habana and cnvrg.io have partnered to bring together the best of both worlds for data scientists and AI developers, offering flexibility, lower costs, and higher productivity for AI training. Enterprises can now easily deploy Gaudi's AI computational power and cost-efficiency with cnvrg.io MLOPs platform.

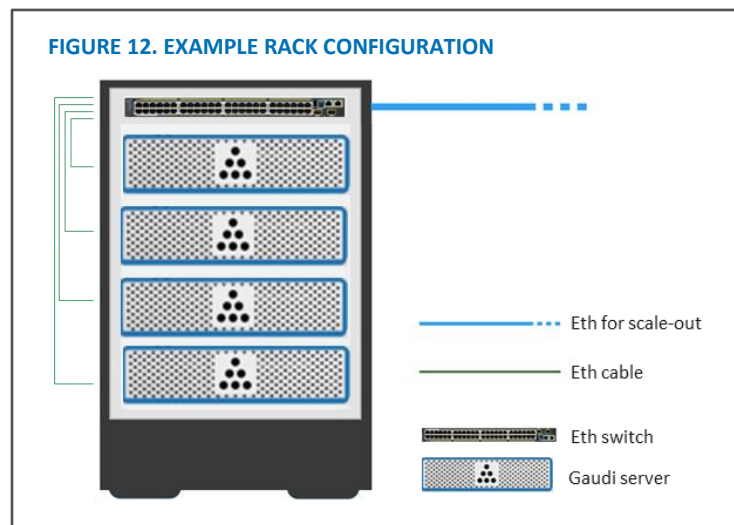
With cnvrg.io, data scientists can deploy more models with drag and drop machine learning pipelines. You can easily run and track experiments, and automate your machine learning from research to production using reusable components and drag-n-drop interface. Getting started with Habana Gaudi on cnvrg.io first requires setting up a Kubernetes cluster for your on-premise Gaudi servers or an Amazon EKS cluster using DL1 EC2 instances. cnvrg.io seamlessly integrates both on-premises and cloud compute resources. The Habana Vault, which hosts the SynapseAI TensorFlow and PyTorch Docker container images, is integrated and available on cnvrg.io *Registries*. You can now bring up a new *Jupyter Workspace*, select the appropriate Gaudi compute and Docker image from the cnvrg.io Habana container registry. You can then get started with the Habana reference models by simply adding the repo location in the cnvrg *Project Settings Git Integration* page. Now you can start a new *Experiment* in cnvrg.io and begin training your model on Gaudi.

VII. RACK LEVEL INTEGRATION

As Gaudi supports standard scaling interfaces, natively, it makes scaling from a single node to racks and cluster that much easier. It is a key differentiator from GPU-based solutions that use proprietary interfaces, and external NIC cards for connecting to network switches. For IT decision makers, avoiding proprietary interfaces in their gear is key for avoiding being “locked” to a single vendor.

Habana has worked with both server, switch, and storage systems partners to make it easy for end-customers to build AI racks and clusters.

The figure below shows a rack-scale configuration with four Gaudi servers connected to a single Ethernet switch at the top of the rack. This switch can be further connected to other racks to form a much larger training pod that can hold hundreds or thousands of Gaudi processors.



Gaudi2 is now available to Habana’s customers, who work on commercializing Gaudi2-based servers for Enterprise customers. Habana will be collaborating with Storage, Switch and OEM partners to build scalable AI appliances with efficient compute, storage and networking to accelerate enterprise AI deployments.



DDN A3I Reference Architecture for Gaudi AI Servers

DDN A3I solutions (Accelerated, Any-Scale AI) are architected to achieve the most from at-scale AI training applications running on Habana Gaudi AI processors. They provide predictable performance, capacity, and capability through integration between DDN AI400X2 appliances and Supermicro X12 Gaudi AI servers. Every layer of hardware and software engaged in delivering and storing data is optimized for fast, responsive, and reliable access. DDN A3I solutions are designed, developed, and optimized in close collaboration between Habana, DDN and Supermicro.

The DDN A3I scalable architecture integrates X12 Gaudi AI servers with DDN AI shared parallel file storage appliances and delivers fully optimized end-to-end AI acceleration on Habana Gaudi AI processors. DDN A3I solutions greatly simplify the deployment of X12 Gaudi AI servers in single server and multi-server configurations, while also delivering performance and efficiency for maximum Habana Gaudi AI processors saturation, and high levels of scalability.

This section describes the components integrated in DDN A3I Solutions with Supermicro X12 Gaudi AI servers.

- DDN AI400X2 Appliance is a fully integrated and optimized shared data platform with predictable capacity, capability, and performance. Every AI400X2 appliance delivers over 90 GB/s throughput and 3M IOPS directly to X12 Gaudi AI servers.
- The Supermicro X12 Gaudi AI server (SYS-420GH-TNGR), powered by Habana's first-generation Gaudi AI Processors, pushes the boundaries of deep learning training and can scale up to hundreds of Gaudi processors in one AI cluster.
- Arista network switches provide optimal interconnect for Habana Gaudi AI processors. For the Gaudi network, the Arista DCS-7060DX4-32 400 Gb/s Ethernet Switch provides 32 ports of connectivity in a 1U form factor. For the Storage and Cluster Management Network, the Arista 7170-32C 100 Gb/s Ethernet Switch provides 32 ports of connectivity in a 1U form factor. For the Management Network, the Arista 7010T Ethernet Switch is a Gigabit Ethernet Layer 3 switch family featuring 54 ports with 48 10/100/1000BASE-T ports, 4 x 10G SFP+ uplink ports. It provides robust capabilities for critical low-intensity traffic like component management.

As general guidance, DDN recommends an AI400X2 appliance for every four X12 Gaudi AI servers. These configurations can be adjusted and scaled easily to match specific workload requirements. An overview of the network architecture is shown in the figure below.

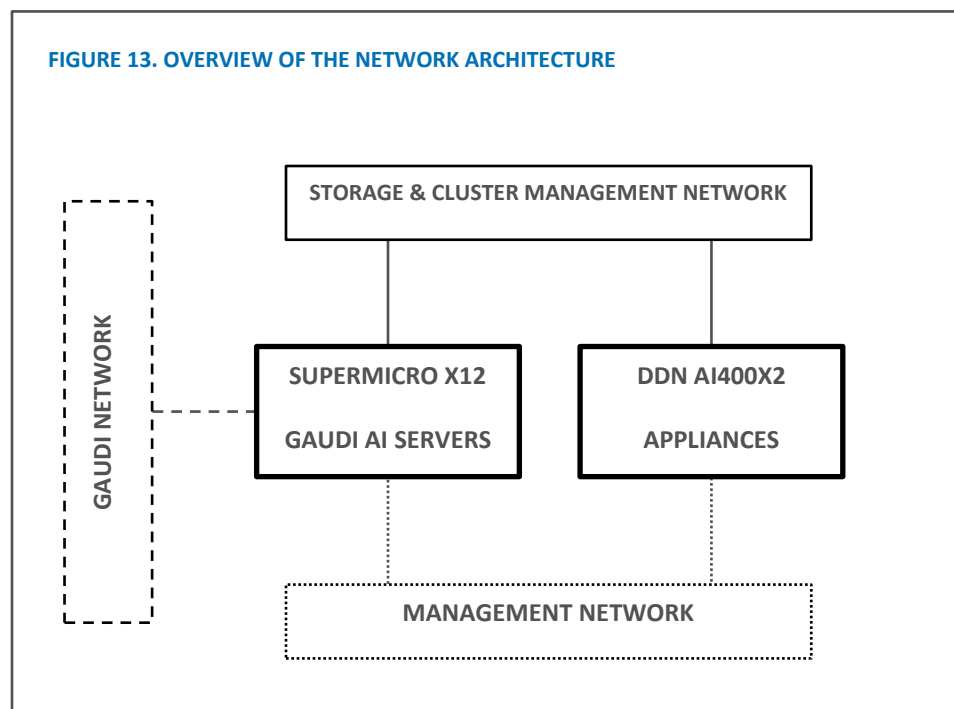
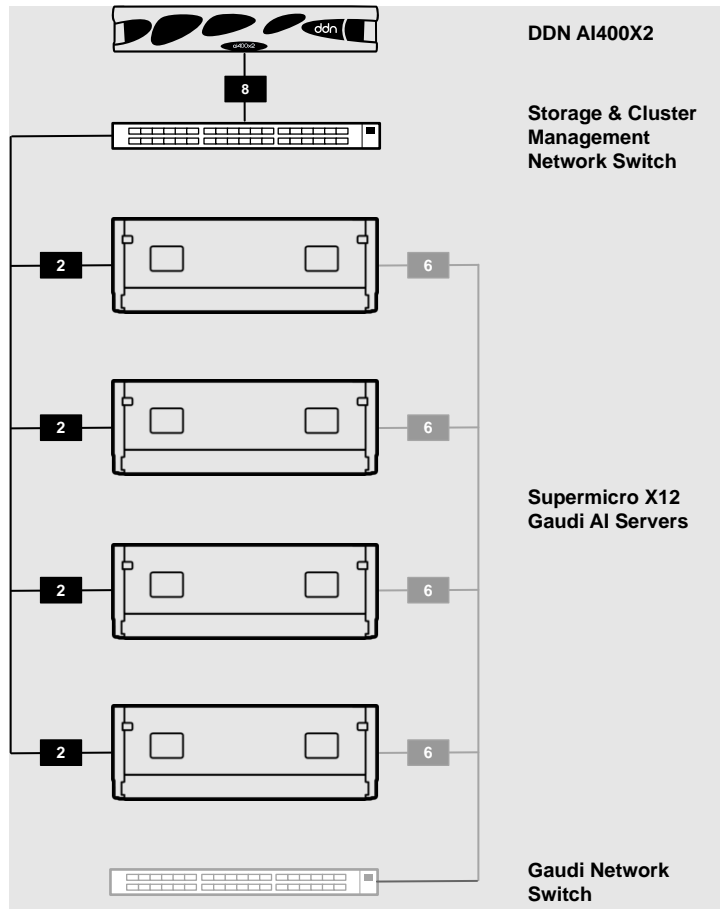


FIGURE 14. DDN A³I REFERENCE ARCHITECTURE WITH FOUR X12 GAUDI AI SERVERS



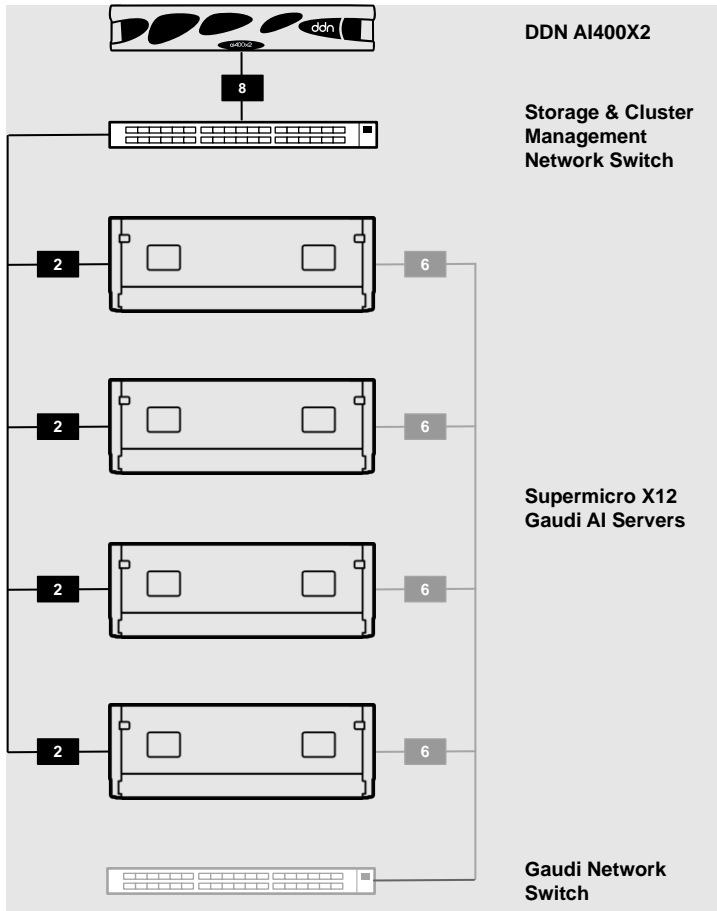


Figure 14 illustrates the DDN A3I architecture quad server configuration. Four X12 Gaudi AI servers are connected to an AI400X2 appliance through a network switch. Every X12 Gaudi AI server connects to the storage & cluster management network switch via two 100 GbE links. The AI400X2 appliance connects to the storage & cluster management network switches via eight 100 GbE links. This ensures non-blocking data communication between every device connected to the network. The multi-path design provides full redundancy and maximum data availability in case a link becomes unavailable.

Additionally, the X12 Gaudi AI servers are connected through a network switch for Gaudi communication. Every X12 Gaudi AI server connects to the Gaudi network switch via six 400 GbE links.



VIII. THE GAUDI®2 OAM CARD

Gaudi2 is offered to system designers in standard OCP OAM 1.1 Mezzanine card form and supports up to 600 TDP Power with passive cooling.

The following table provides its key interfaces

TABLE 2. HL-225H KEY INTERFACES

Interface	Description
Host i/f	X16 PCIe Gen3/4
Networking: Card to Card & Scale-out	48 x 56Gb PAM4 SerDes Links, Supporting 24x100GbE RoCE v2
JTAG	In-field CPLD programming
UART	Low level debug & BMC access
I2C Master	On/Off board Peripherals
I2C Slave / SMBUS	BMC control and monitoring interface



IX. GAUDI®2 HLBA-225 BASEBOARD

The HLBA-225 is another product, inspired by OCP, and offered for simplifying system design for Gaudi2-based system designers. It supports eight Gaudi2 mezzanine cards which are passively interconnected on its PCB in a non-blocking all-to-all configuration, using from each card 21 NICs (3 ports to every other of the 7 cards), as well as routing the 3 remaining NICs from every Gaudi2 card (3x8=24) to the six on-board QSFP-DD connectors, for scaling-out.

The baseboard has standard interface/connectors to the HIB (Host Interface Board), which allows the system designer the customization to design to specific needs and the flexibility to build systems of choice with a different ratio of CPUs to accelerators for different kind of topologies and applications.

TABLE 3: HLBA-225 KEY PROPERTIES

Feature	Description
OAM support	<ul style="list-style-type: none">• All to all connectivity for 8 Gaudi2 HL-225H cards• OAM powered by 54v and 12v• x16 PCIe Gen4 host interface per OAM• 8 X dual B2B connectors
Baseboard to HIB (Host Interface Board) Interface	<ul style="list-style-type: none">• 8 X 16 PCIe Gen 4 connectors• Power: 12V, 54V• Side band signals: I2C, Reset, reference clocks, JTAG, UART, SGMII, USB• Eight Amphenol connectors: 2x 160P (10131762-101LF) + 6 x 112P (10137002-101LF)
Networking: Card to Card & Scale-out	<p>Per OAM: 24x 100GbE PAM4 SerDes Links split into:</p> <ul style="list-style-type: none">• 21 x 100GbE for OAMs-to-OAMs connections• 3 x 100GbE for scale-out <p>Total Scale-out:</p> <ul style="list-style-type: none">• 8 x 3 x 100GbE = 2.4TbE connected to 6 QSFP-DD ports
PCB dimension	<ul style="list-style-type: none">• 585mm x 417mm x 4.6mm

Block Diagram and Main Components

HLBA-225 has the following main components:

- 8 X dual B2B connectors for the HL225H Mezzanine boards

- High speed connectors for x16 PCIe interconnect to HIB
- 2 CPLDs
- Power and reset control
- JTAG distribution to the mezzanines
- LED indications
- 6x QSFP-DD connectors (4x400G using 56G PAM 4 SerDes)
- 3x PHY retimers
- 8x PCIe retimers
- USB connectors for Debug Management and control
- RJ45 connector for BMC management

FIGURE 15. HLBA-225 KEY COMPONENTS

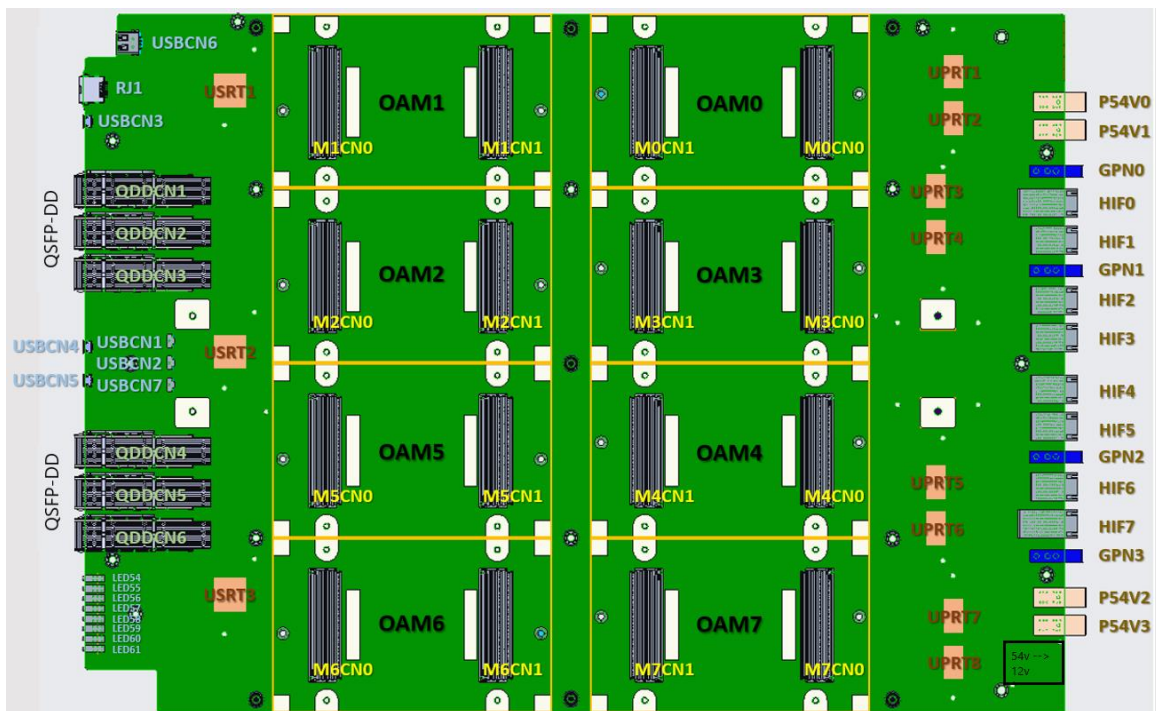
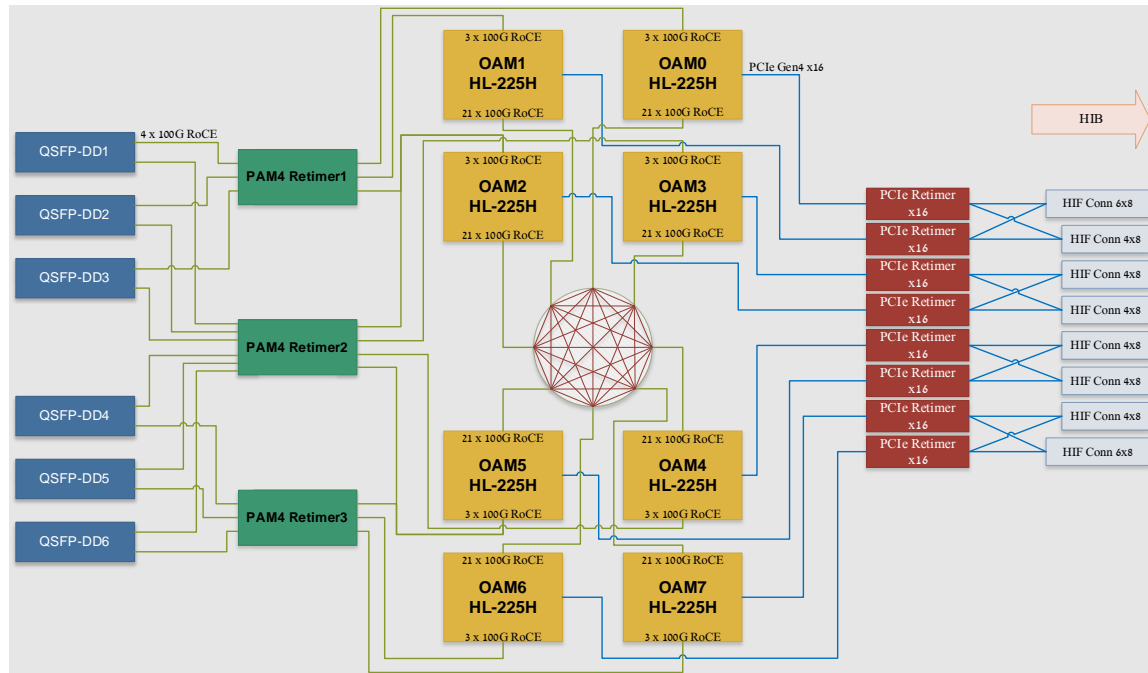


Figure 16. HLBA-225 High Speed Block Diagram





X. HLS-GAUDI®2 Server

The HLS-Gaudi®2 system is a high-performance deep-learning server, incorporating a dual socket Xeon host subsystem and 8 Gaudi2 accelerators, which supports scaling out using 24x100GbE RDMA ports.

HLS-Gaudi®2 has the following main features:

TABLE 4. HLS-GAUDI2 KEY FEATURES

Feature	Description
System Dimension	<ul style="list-style-type: none">• 19”
CPU head node	<ul style="list-style-type: none">• 2* INTEL Xeon Ice Lake CPU• 32* DDR4 DIMM• 2* NIC
HIB	<ul style="list-style-type: none">• 2* PCIe switch• BMC + peripheral
Base Board	<ul style="list-style-type: none">• HLBA-225• fully connected topology• 6x QSFP-DD connectors (4x400G using 56G PAM 4 SerDes)
OAM	<ul style="list-style-type: none">• 8* Habana Gaudi®2
	<ul style="list-style-type: none">• 4* PCIe Gen 4 U.2 NVME SSD
PSU	<ul style="list-style-type: none">• 4 (3+1) 4 kW PSU

Figure 17. HLS-Gaudi2 System Layout

