SC23
Denver, CO | i am hpc.

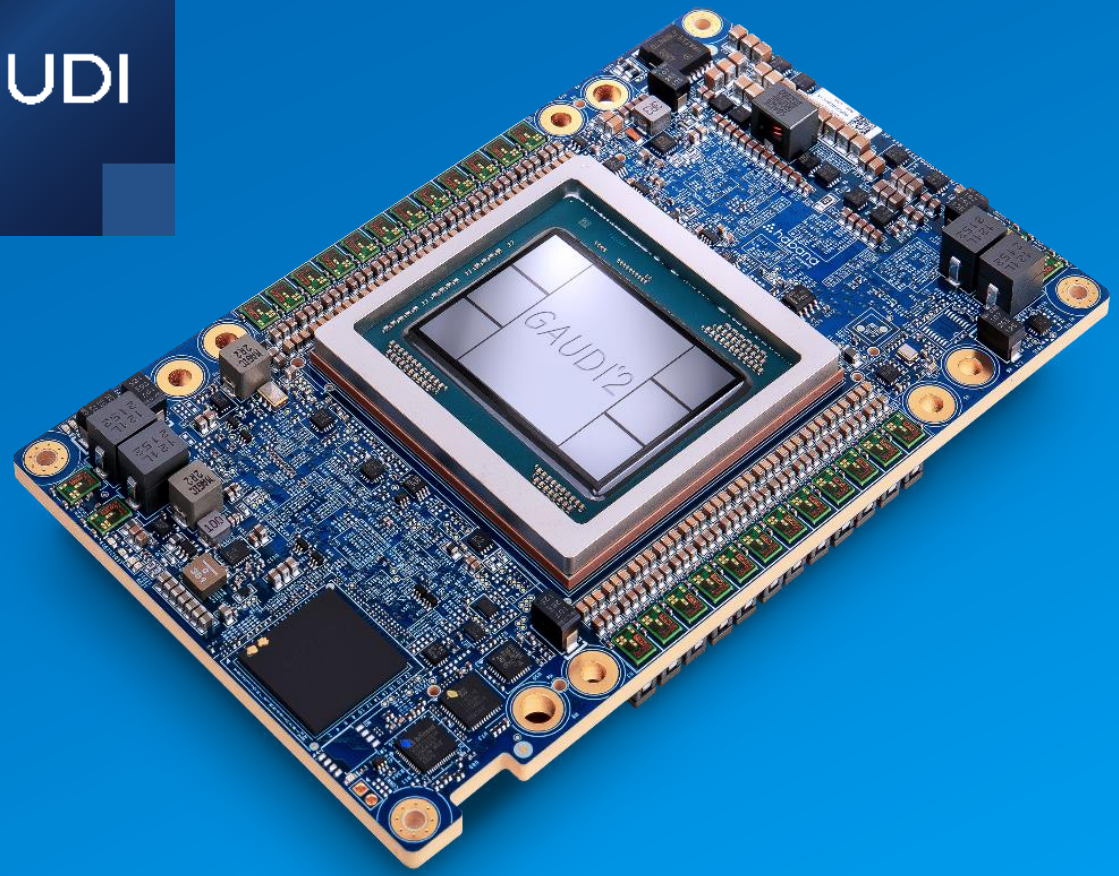
Programming Novel AI Accelerators for Scientific Computing Intel® Gaudi®2 AI Accelerator

Leon Tran (ltran@habana.ai)

www.habana.ai

Architected for deep learning performance and efficiency

intel
GAUDI



Matrix Multiplication Engine &
24
Tensor Processor Cores

96 GB
On-Board HBM2e

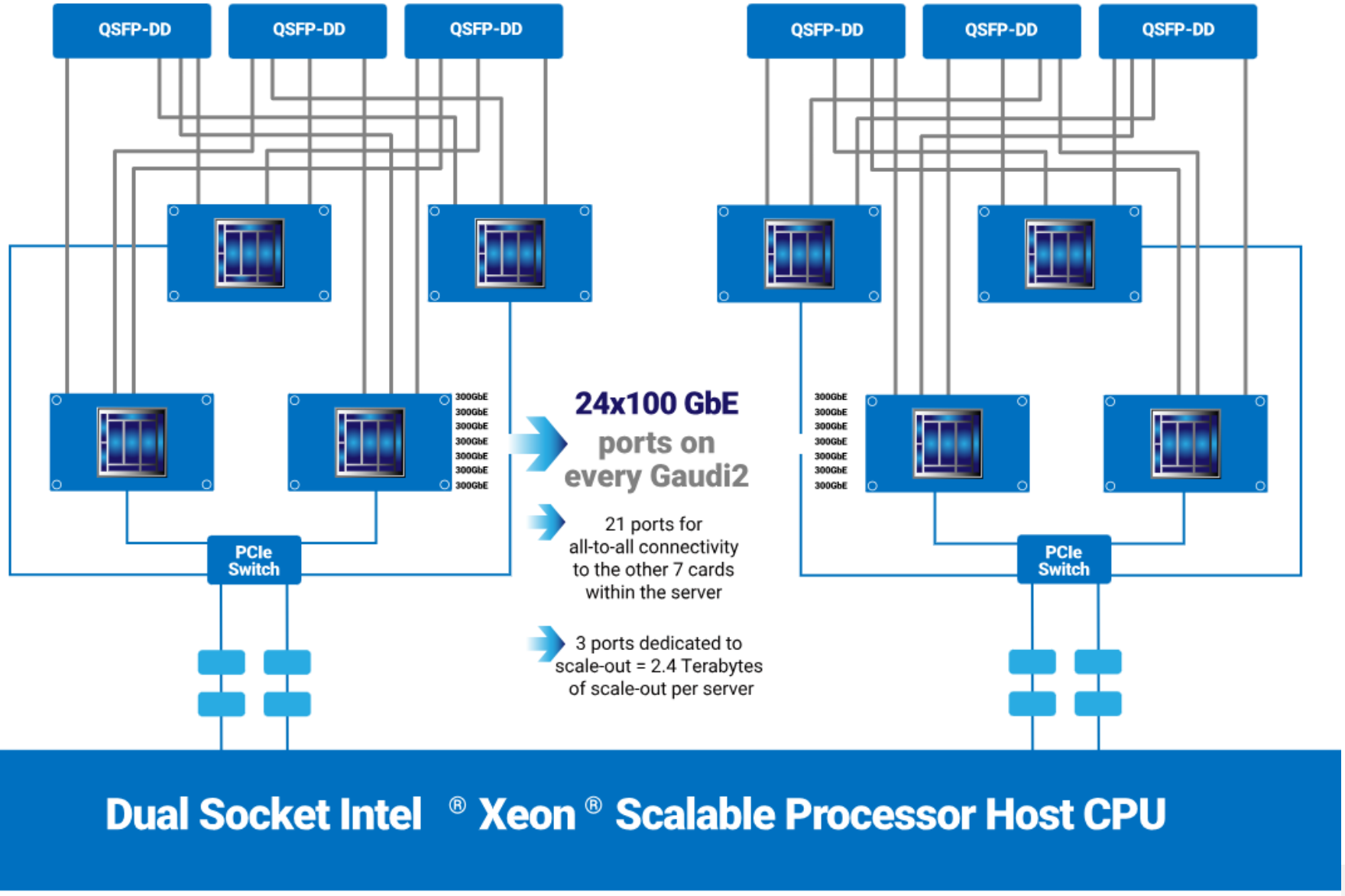
7nm
Process Technology

24
Integrated 100G Ethernet ports

48 MB
SRAM

Purpose Built for Deep Learning Acceleration at Scale

Intel® Gaudi®2 Server for Flexible and Efficient Scalability



Gaudi HL-225H

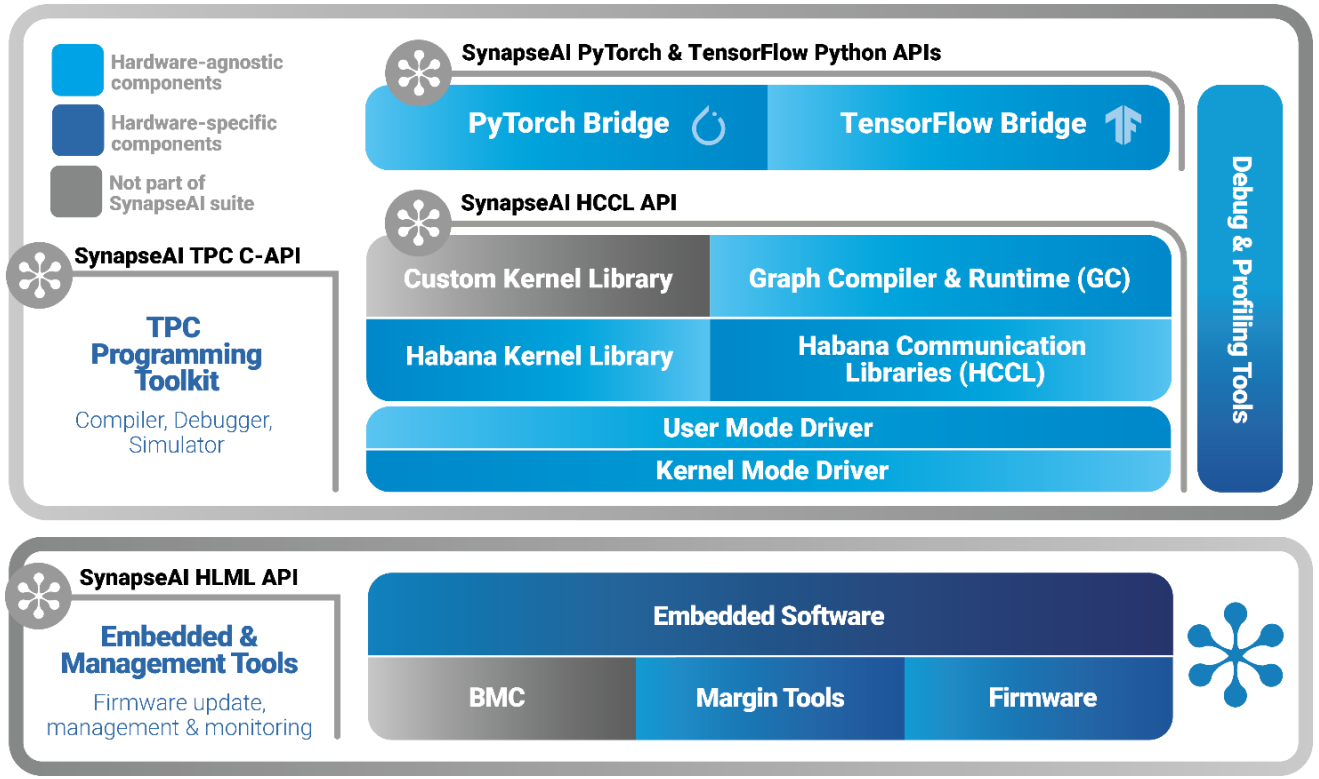
1x100GbE

PCIe

All-to-All Connectivity
21x100G Eth

SynapseAI Software:
Optimized for Intel® Gaudi® Performance and Ease of Use

SynapseAI®



- Shared software suite for training and inference
- Start running on Intel Gaudi accelerators with minimal code changes
- Integrated with PyTorch and TensorFlow
- Rich library of performance-optimized kernels
- Advanced users can write their custom kernels
- Docker container images and Kubernetes orchestration
- Habana Developer Site & HabanaAI GitHub
- Habana Developer Forum

Model Migration Steps - Paths

habana® An Intel Company GitHub

Using Habana Model References GitHub
Fully vetted starting point to validate existing performance or use examples to innovate.

Using Hugging Face
Start with existing examples or use Optimum Habana library with any transformer model

GitHub

Using Public Model
Use the GPU Migration Toolkit or perform Manual Migration

PyTorch Manual Migration

1 Pre Work: Removing CUDA calls

2

```
import torch

# Import Habana Torch Library
import habana_frameworks.torch.core as htcore

# neural network model
class SimpleModel(nn.Module):
```

3

```
# training loop
def train(net,criterion,optimizer,trainloader,device):
...
        loss.backward()
        htcore.mark_step()

        optimizer.step()
        htcore.mark_step()

def main():
...
    # Target the Gaudi HPU device
    device = torch.device("hpu")
```

Autocast For Mixed Precision

4

```
with torch.autocast(device_type="hpu", dtype=torch.bfloat16):    output = model(input)
        loss = loss_fn(output, target)
loss.backward()
```

Distributed Training Setup Example

5

```
import habana_frameworks.torch.distributed.hccl
torch.distributed.init_process_group(backend='hccl')

...
# Use with PyTorch DDP Hook
ddp_model = DDP(model)

(model) loss_fn = nn.MSELoss()
optimizer = optim.SGD(ddp_model.parameters(), lr=0.001)

optimizer.zero_grad()
outputs = ddp_model(torch.randn(20, 10).to(device))
```

GPU Migration Toolkit

GPU Migration Toolkit maps specific API calls from Python libraries and modules like:

torch.cuda

Torch API w/ GPU related parameters like: torch.randn(device="cuda")

Apex. (check Limitations)

pynvml

The GPU Migration toolkit is preinstalled.

GPU Migration Logging allows investigation on what was changed

Logging feature can be enabled by setting the GPU_MIGRATION_LOG_LEVEL environment variable

PyTorch Migration with GPU Toolkit

1 Pre Work: Removing CUDA calls

2

```
import torch
# Import Habana Torch Library
import habana_frameworks.torch.gpu_migration
import habana_frameworks.torch.core as htcore

# neural network model
class SimpleModel(nn.Module):
```

3

```
# training loop
def train(net,criterion,optimizer,trainloader,device):
...
        loss.backward()
        htcore.mark_step()

        optimizer.step()
        htcore.mark_step()

def main():
...
    # Target the Gaudi HPU device
    device = torch.device("hpu")
```

Autocast For Mixed Precision

4

```
with torch.autocast(device_type="hpu", dtype=torch.bfloat16):    output = model(input)
        loss = loss_fn(output, target)
loss.backward()
```

Just 2 steps and go!

Distributed Training Setup Example

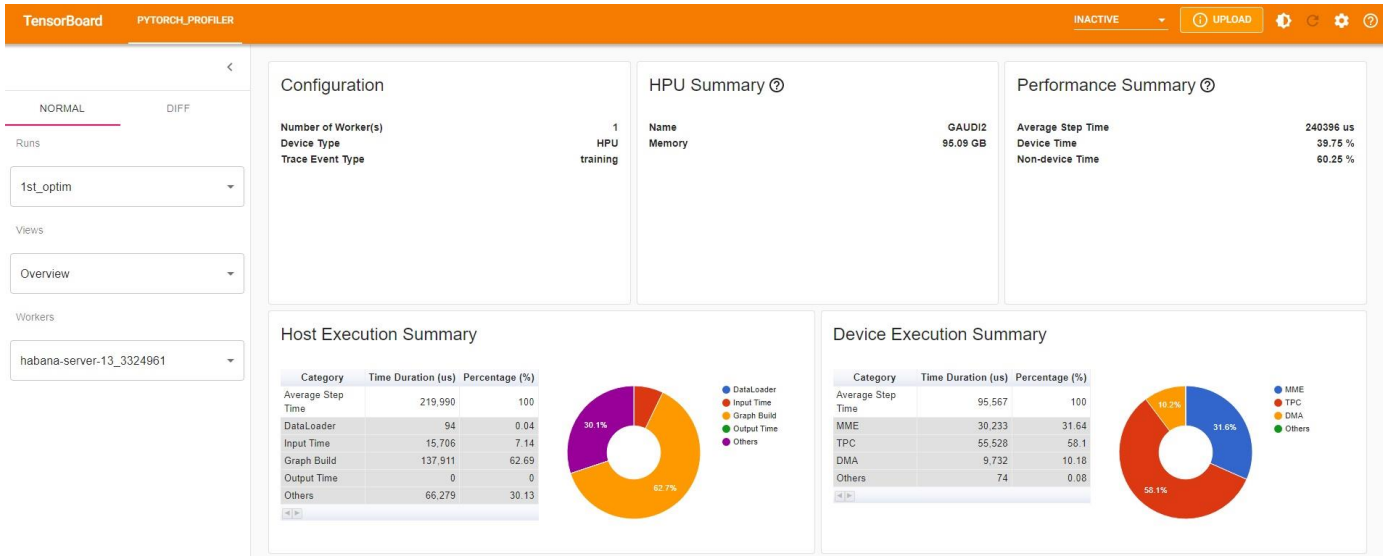
5

```
import habana_frameworks.torch.distributed.hccl
torch.distributed.init_process_group(backend='hccl')

...
# Use with PyTorch DDP Hook
ddp_model = DDP(model)

(model) loss_fn = nn.MSELoss()
optimizer = optim.SGD(ddp_model.parameters(), lr=0.001)

optimizer.zero_grad()
outputs = ddp_model(torch.randn(20, 10).to(device))
```

Profiling

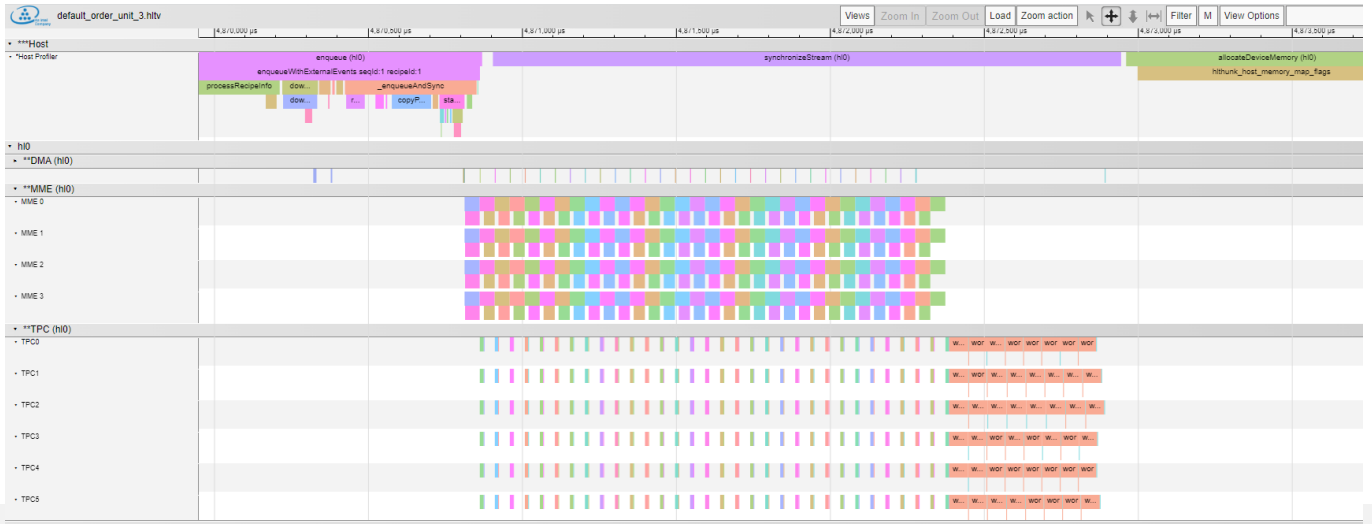
PT Tensorboard

- HPU Overview
- HPU Kernel View
- HPU Memory View
- TraceViewer
- Recommendations for HPU Optimization



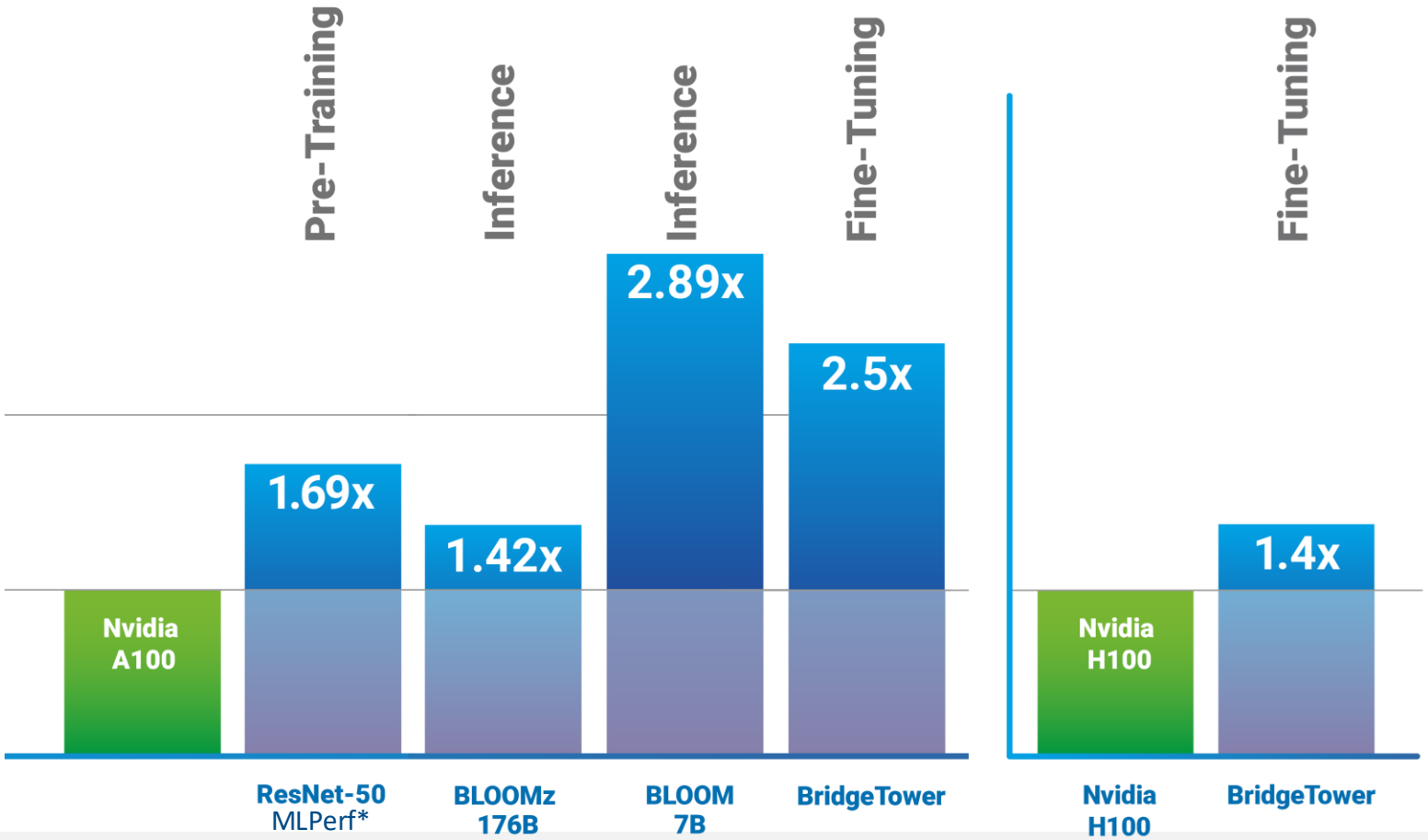
SynapseAI Profiling

- Advance HPU level Debug
- Upload .hltv files to https://hltv.habana.ai/ or Perfetto UI
- Host and HPU analysis



Hugging Face Evaluations of Intel® Gaudi®2 Performance vs. A100 and H100

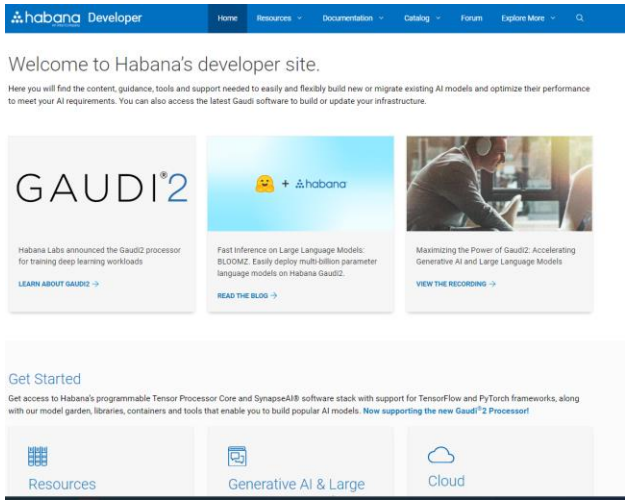
Visit https://habana.ai/habana-claims-validation
for workloads and configurations. Results may vary.
https://huggingface.co/blog/habana-gaudi-2-bloom
https://huggingface.co/blog/bridgetower
MLPerf Benchmark



Intel® Gaudi® Developer Platform

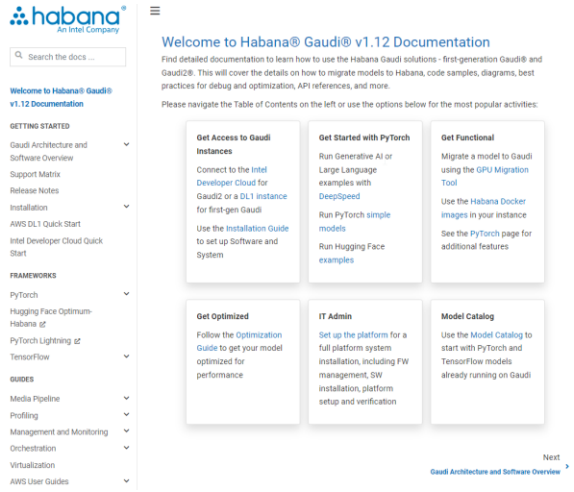
Habana Developer Portal

- Performance
- Catalog
- Tutorial
- News



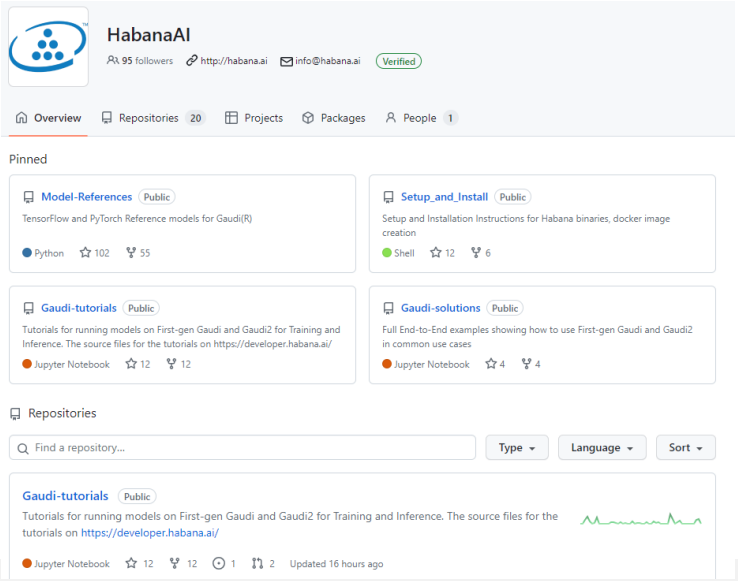
Habana Documentation

- Setup & Install
- User Guides
- Migration



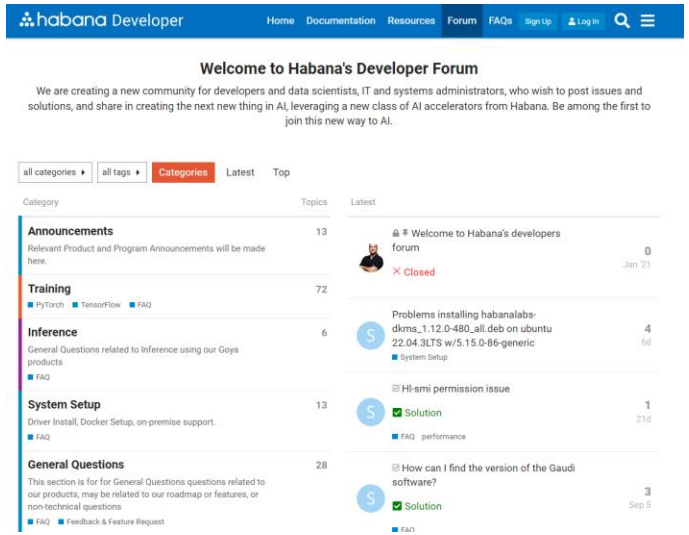
GitHub

- Model References
- Optimum Habana
- Tutorials



Service Desk & Support Forum

- Announcements
- Community Support



Thank you

Q&A


intel