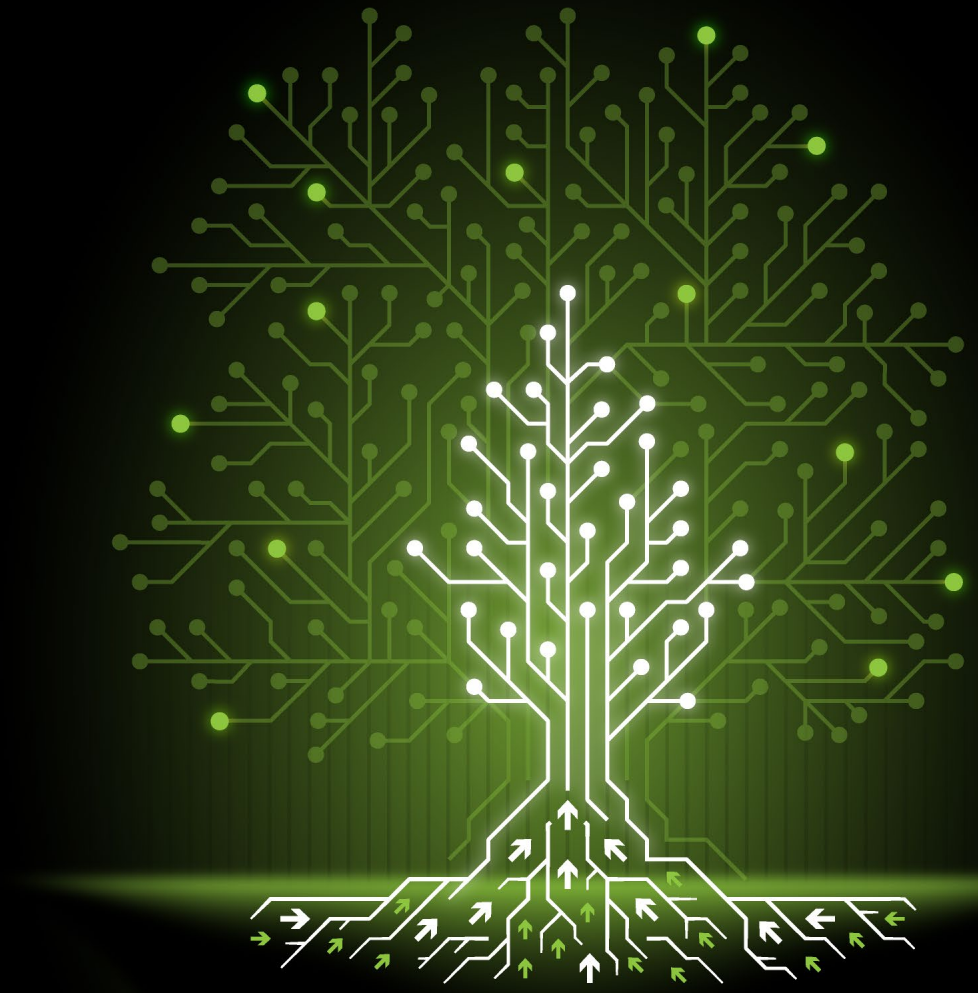
Dan Rabinovitsj

Vice President, Infrastructure

Meta



OCP GLOBAL SUMMIT

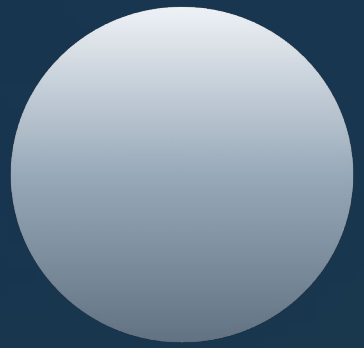
Opening AI Infrastructure

Ushering In The Age Of GenAI

Meta

AI workloads are fast evolving…

Recommendation-DLRM
Circa: 2022



…exponential changes

Recommendation-DLRM
Circa: 2022

Llama-65B
Circa: 2023

...exponential changes

Recommendation-DLRM
Circa: 2022

Llama-65B
Circa: 2023

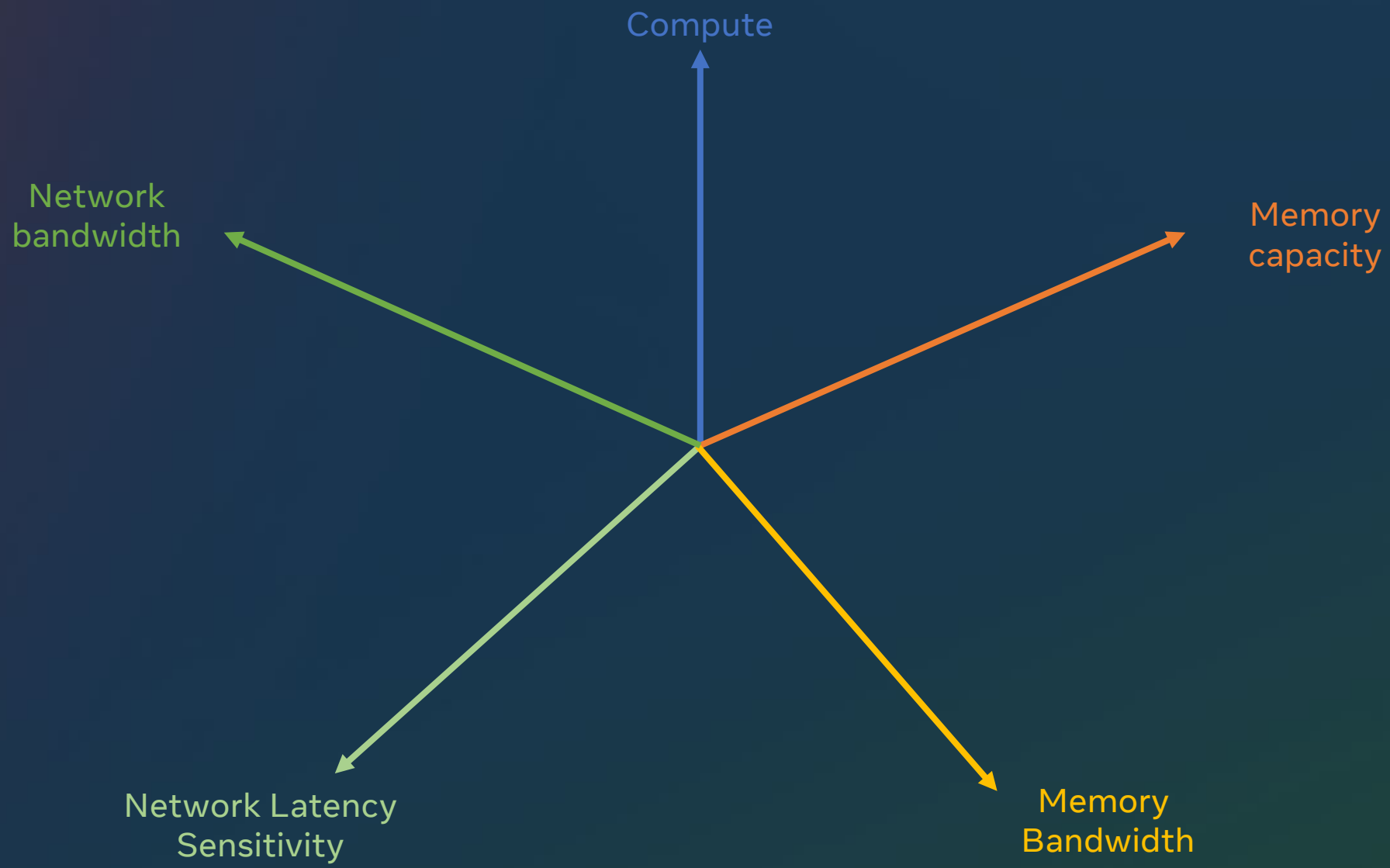
Total Compute (PF/s)	1	400
Memory Capacity (TB)	1	10
Training Scale (GPUs)	1x	20x

and we are not done yet...

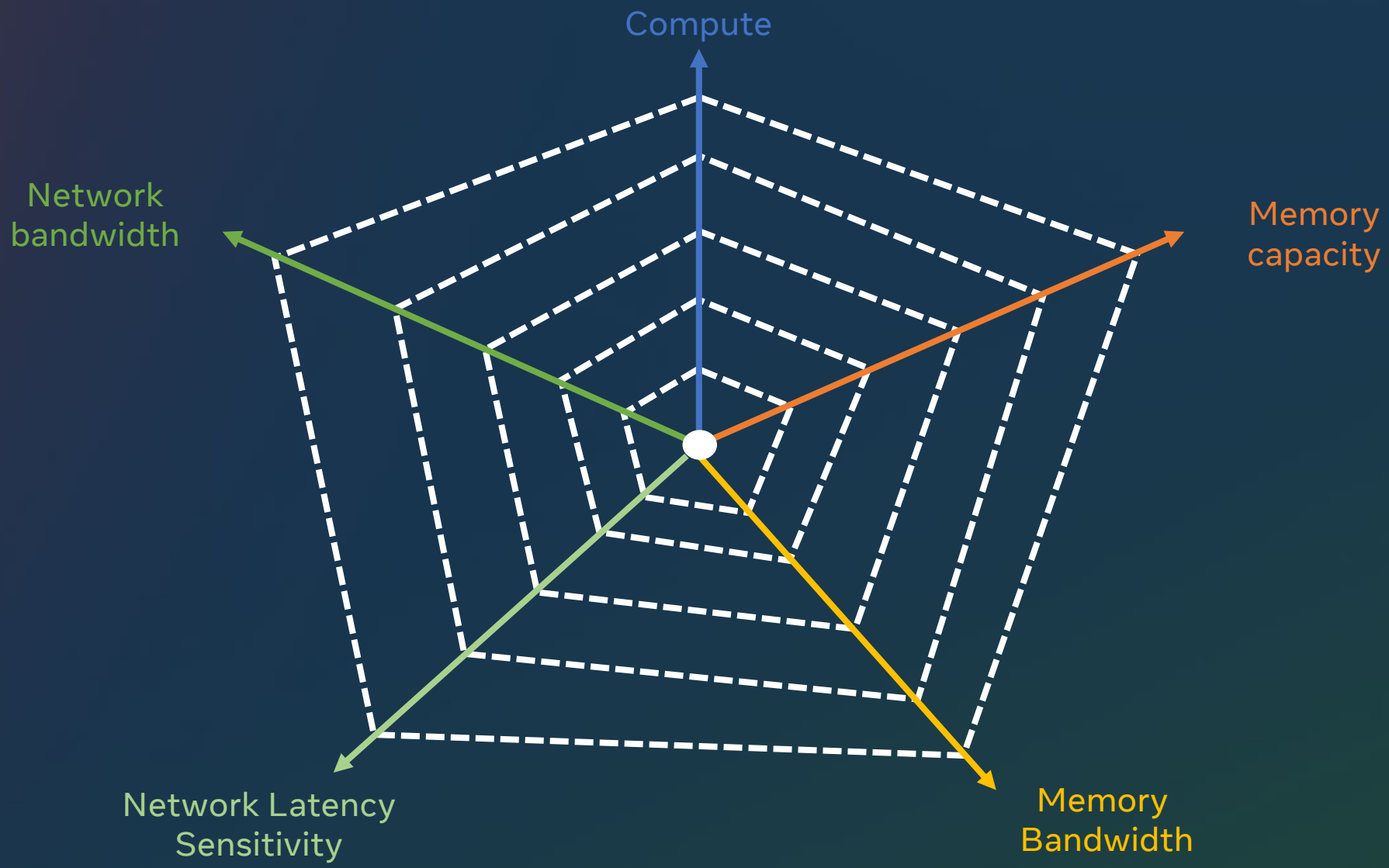
Llama-65B
Circa: 2023

Llama-XX
Circa: 202?

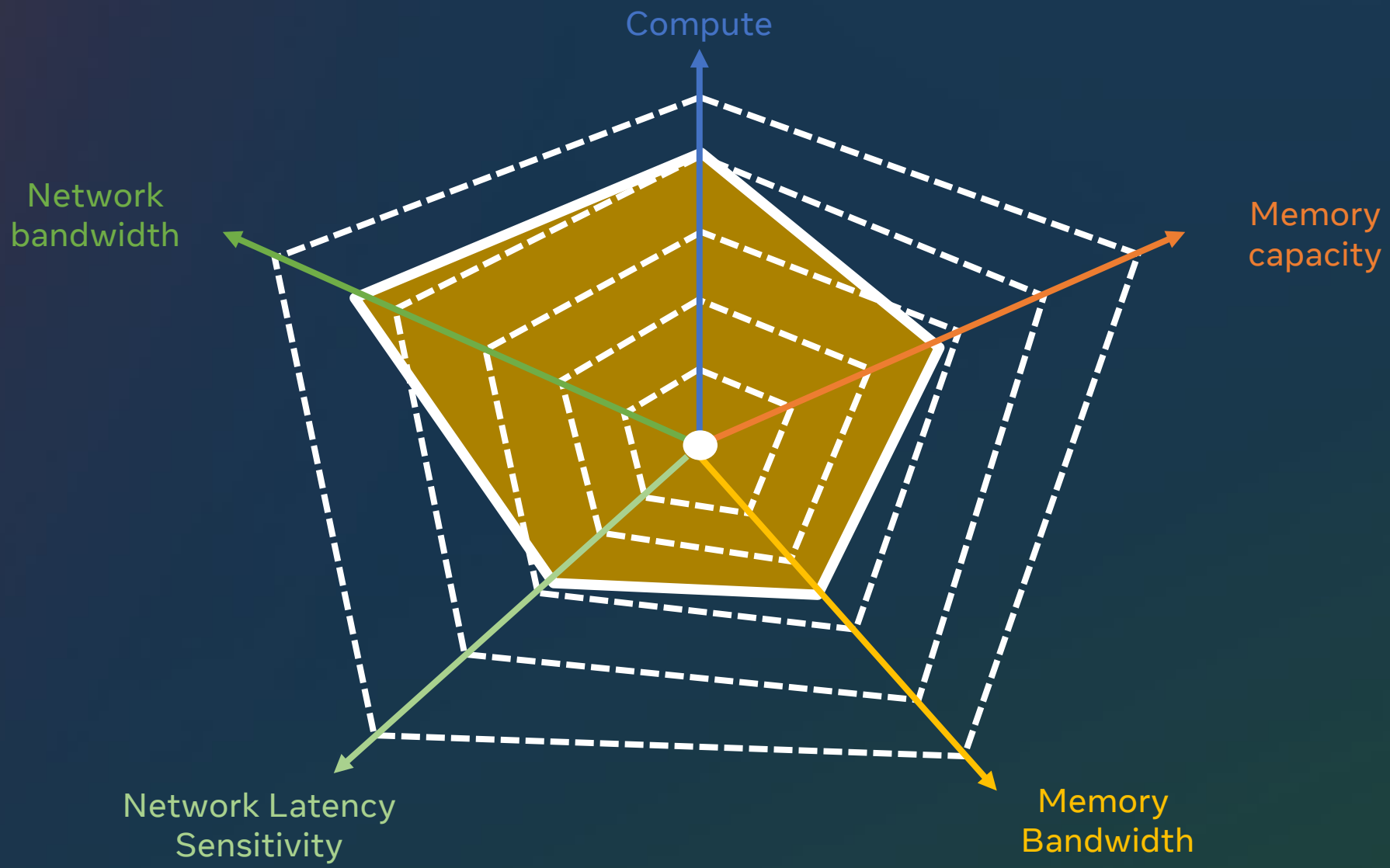
AI Systems Require Multi-Dimensional Design



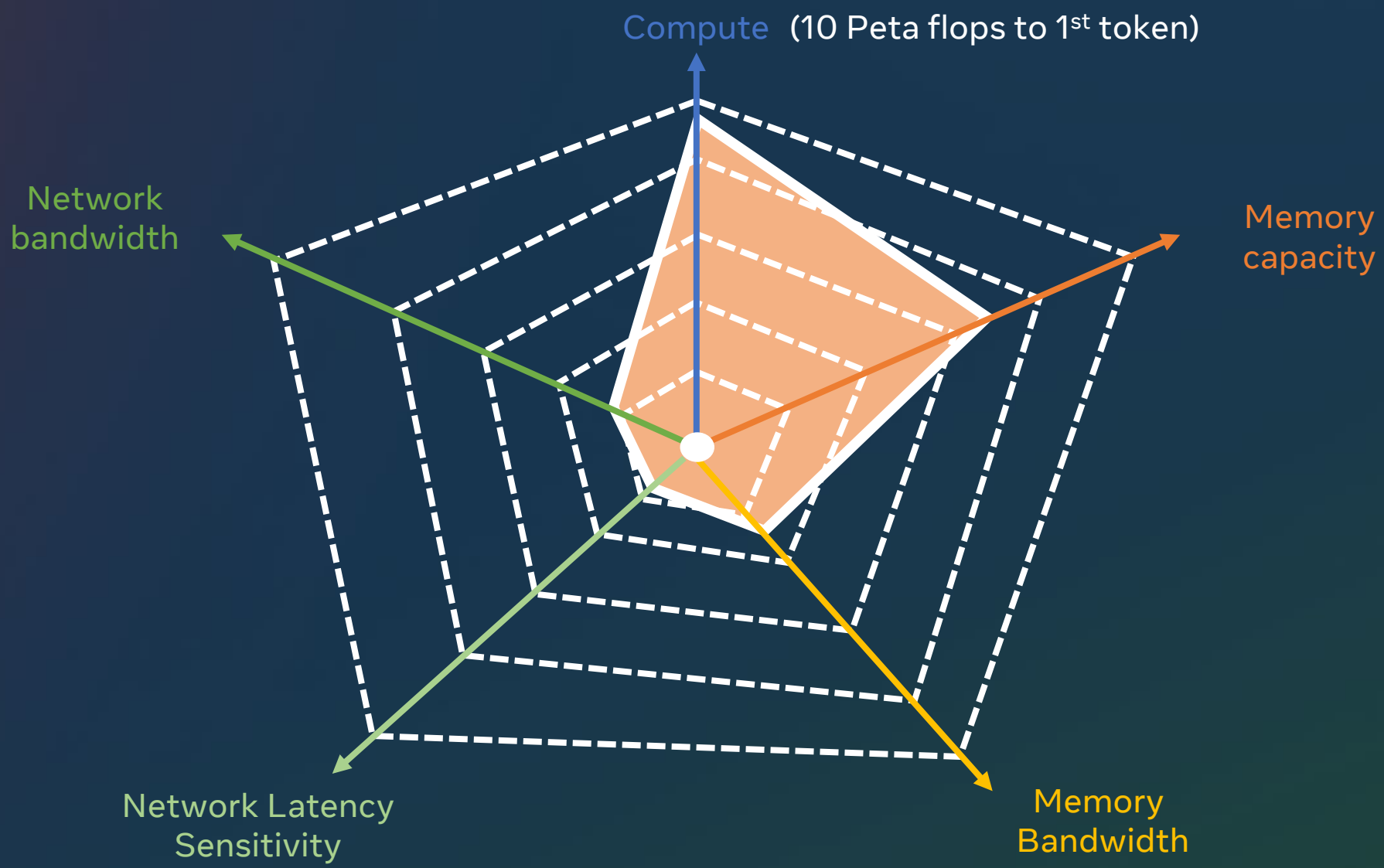
AI workloads impose diverse demands



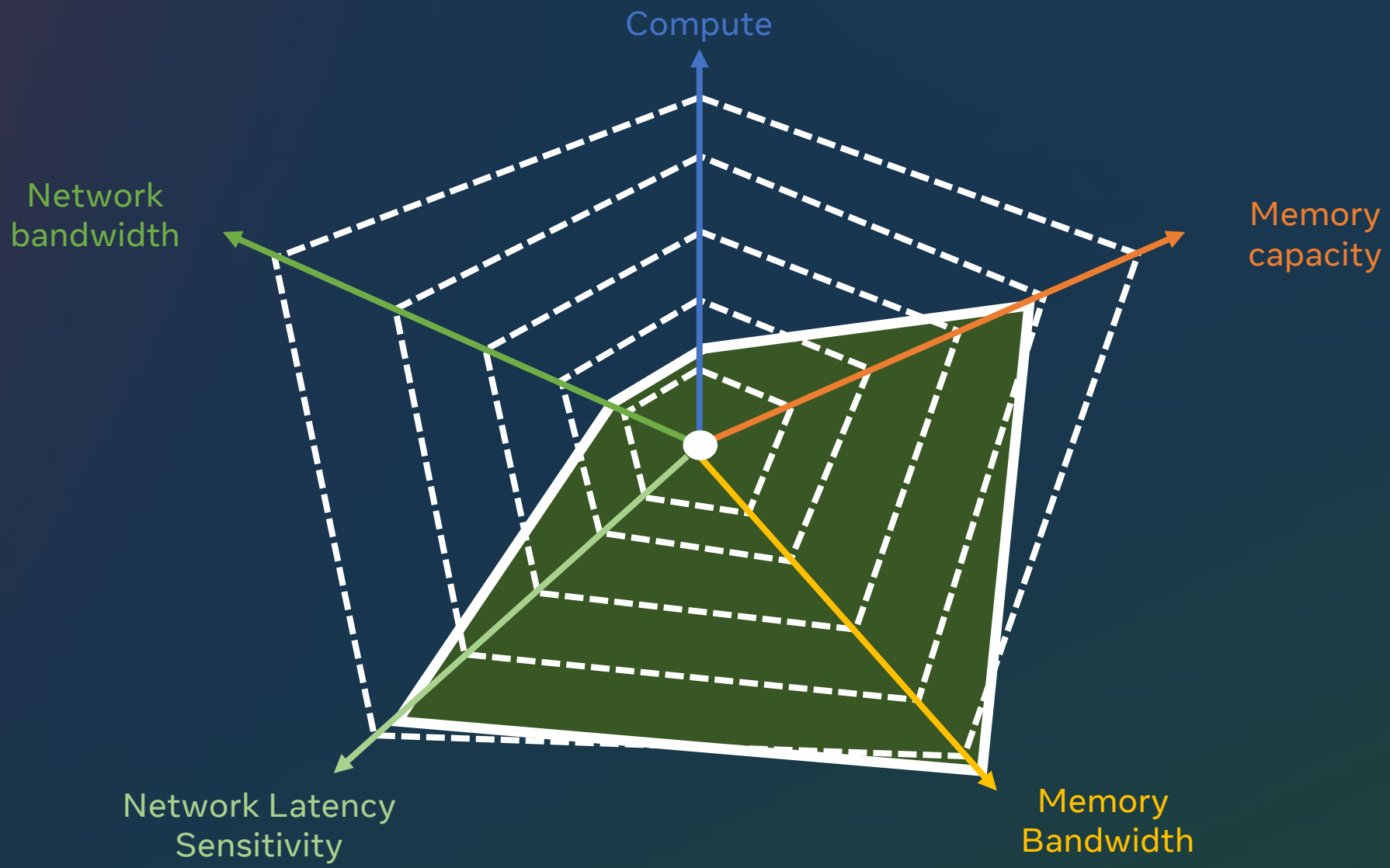
Large Language Models (LLMs) Training



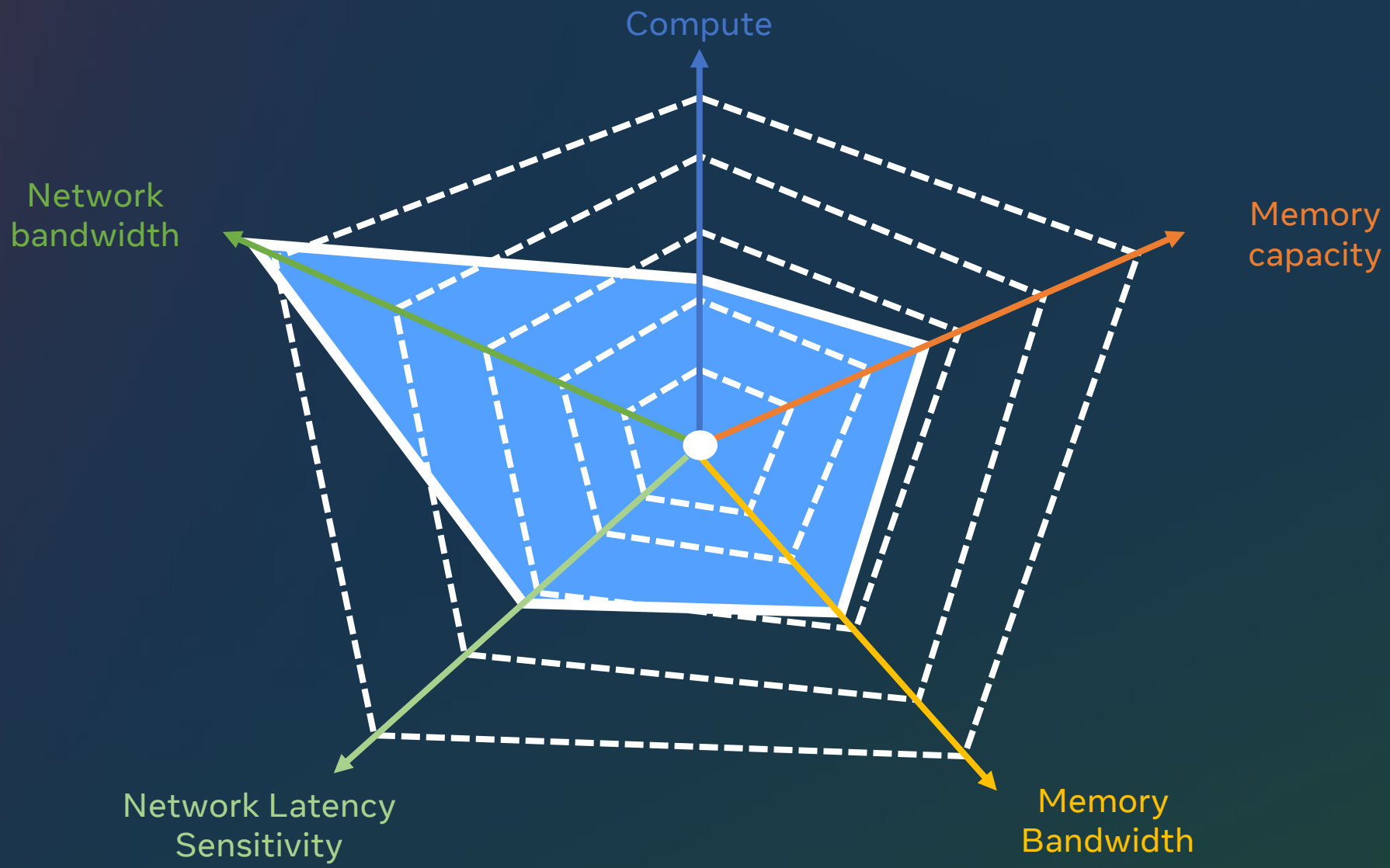
LLMs Inference Prefill



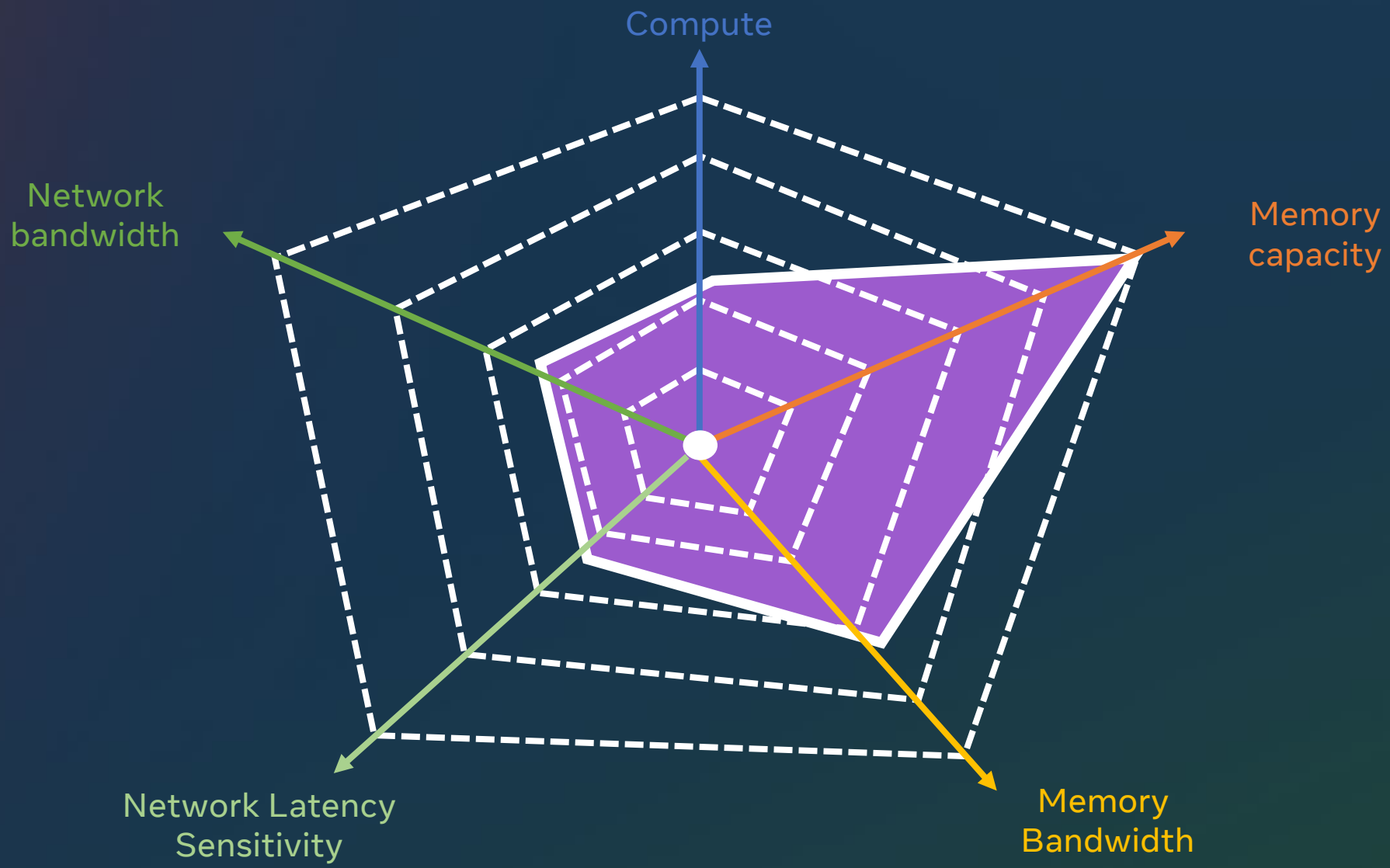
LLMs Inference Decode



Ranking and Recommendation Training

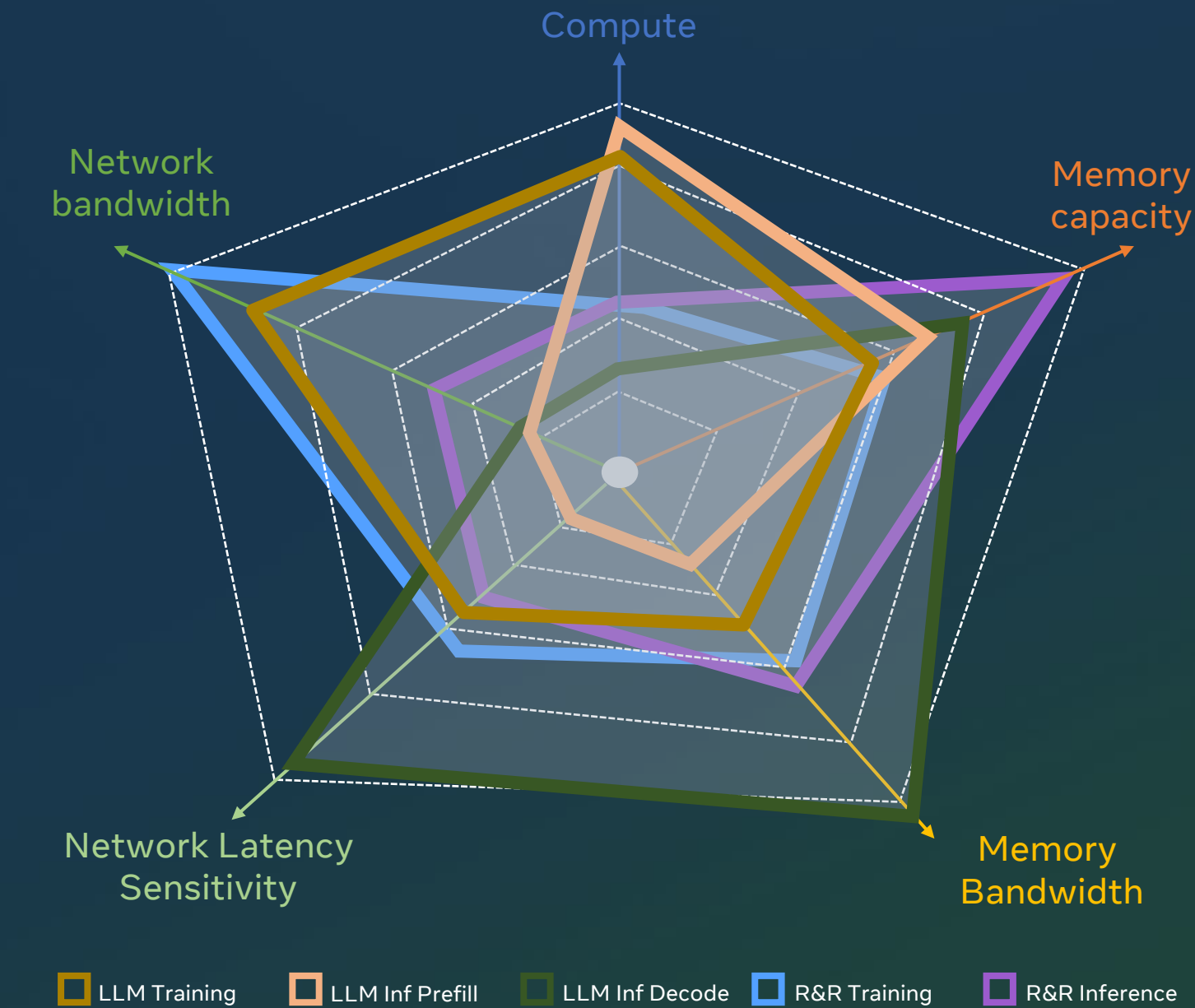


Ranking and Recommendation Inference



Workload Diversity Continues

- Difficult to serve all classes of models with a single system design point
- New models & parallelism techniques put unexpected pressures on AI systems
- The next frontier of innovation is in software/hardware co-design



AI Systems tuned for diverse applications

All images are generated using Meta EMU

The Big Question

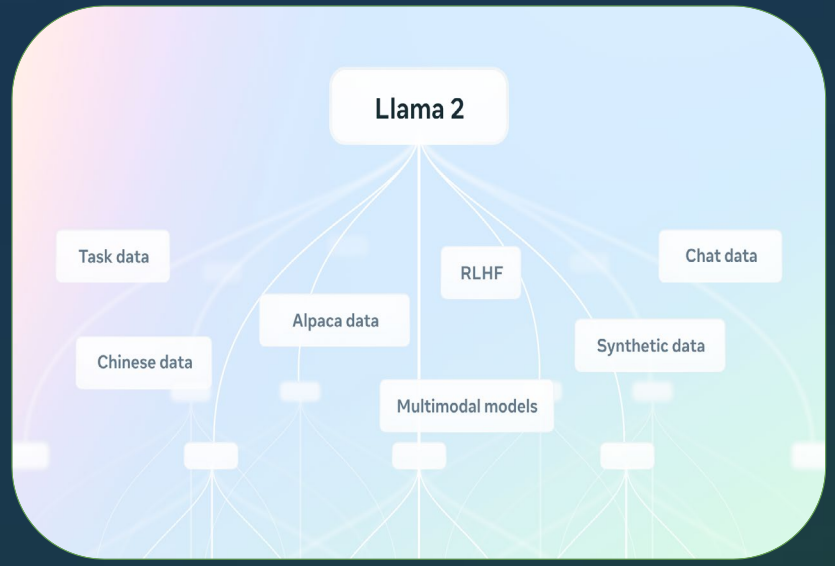
Is it realistic to open finely-tuned, vertically-integrated AI systems?

AI for All: The open model is already out there

AI for All: The open model is working!

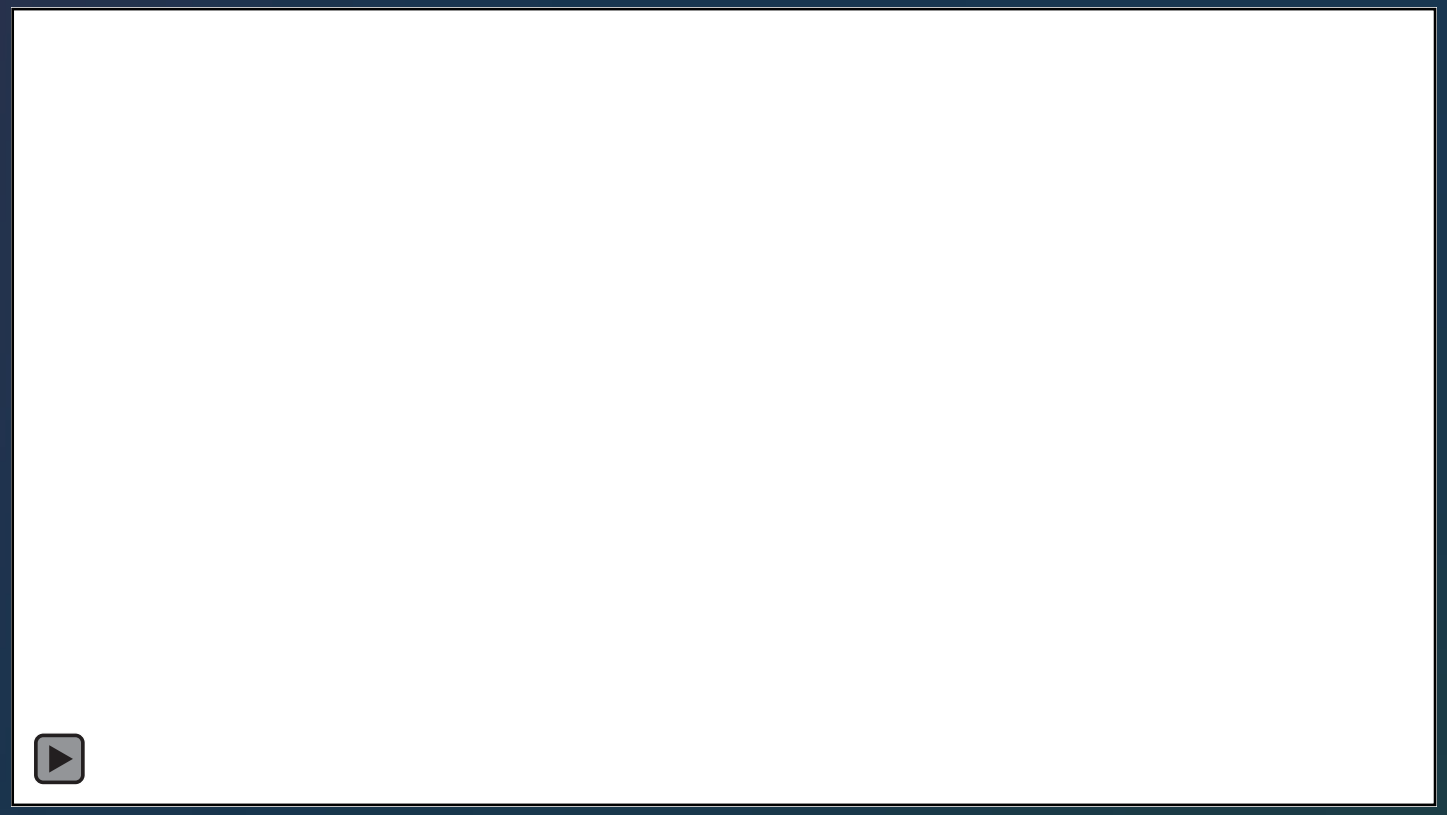
```python
import torch
from torch.autograd import Variable
import torch.nn as nn
import torch.nn.functional as F


class Net(nn.Module):

    def __init__(self):
        super(Net, self).__init__()
        # 1 input image channel, 6 output channels, 5x5 square convolution
        # kernel
        self.conv1 = nn.Conv2d(1, 6, 5)
        self.conv2 = nn.Conv2d(6, 16, 5)
        # an affine operation: y = Wx + b
        self.fc1 = nn.Linear(16 * 5 * 5, 120)
        self.fc2 = nn.Linear(120, 84)
        self.fc3 = nn.Linear(84, 10)

    def forward(self, x):
        # Max pooling over a (2, 2) window
        x = F.max_pool2d(F.relu(self.conv1(x)), (2, 2))
        # If the size is a square you can only specify a single number
        x = F.max_pool2d(F.relu(self.conv2(x)), 2)
        x = x.view(-1, self.num_flat_features(x))
        x = F.relu(self.fc1(x))
        x = F.relu(self.fc2(x))
        x = self.fc3(x)
        return x

    def num_flat_features(self, x):
        size = x.size()[1:]  # all dimensions except the batch dimension
        num_features = 1
        for s in size:
            num_features *= s
        return num_features


net = Net()
print(net)
```

PyTorch



OPEN Compute Project®

Our AI journey began with Grand Teton



Grand Teton GPU Clusters in Production

Building Open AI Infrastructure

Common

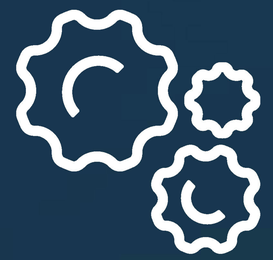
Chassis & Rack Architecture

Standard

Liquid Cooling Designs & Blind Mate Connectors

Open

Integration, Validation and Test Playbooks



Universal

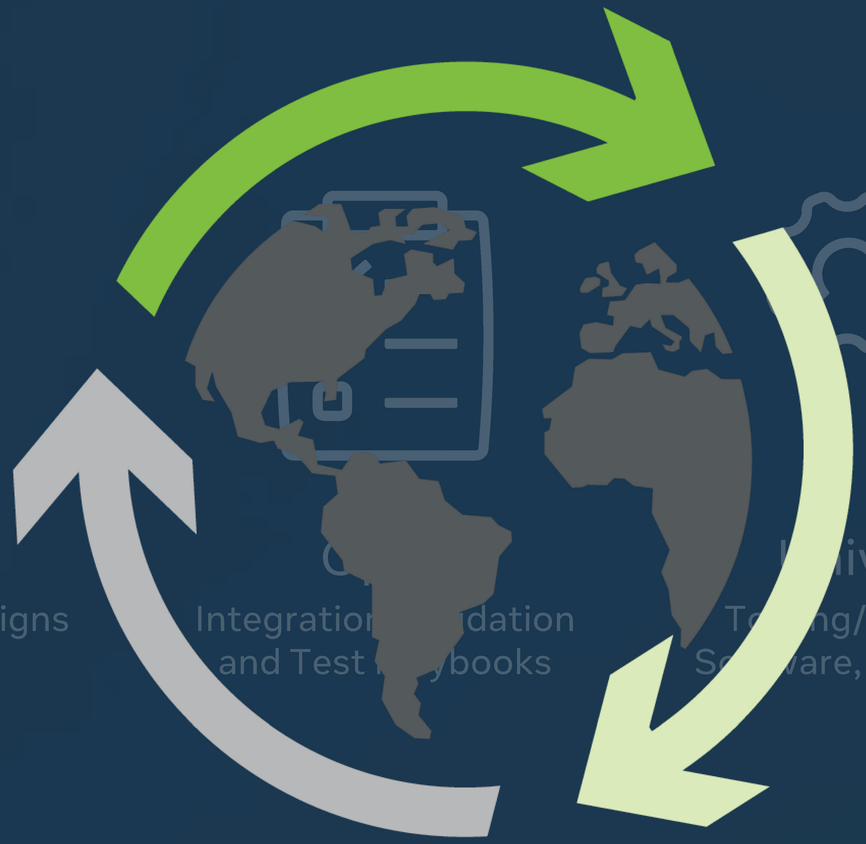
Tooling/Telemetry, Software, and Support

Unified

Open backend Fabric options

TIME

Building Open AI Infrastructure



Common
Chassis & Rack Architecture

Standard
Liquid Cooling Designs & Blind Mate Connectors

Universal
Open backend Fabric options

Unified

TIME

Takeaways

AI Infrastructure complexity is growing

OCP is needed for emerging AI Systems

AI for All requires an open community approach


Meta