Distributed AI Without the Infrastructure Tax

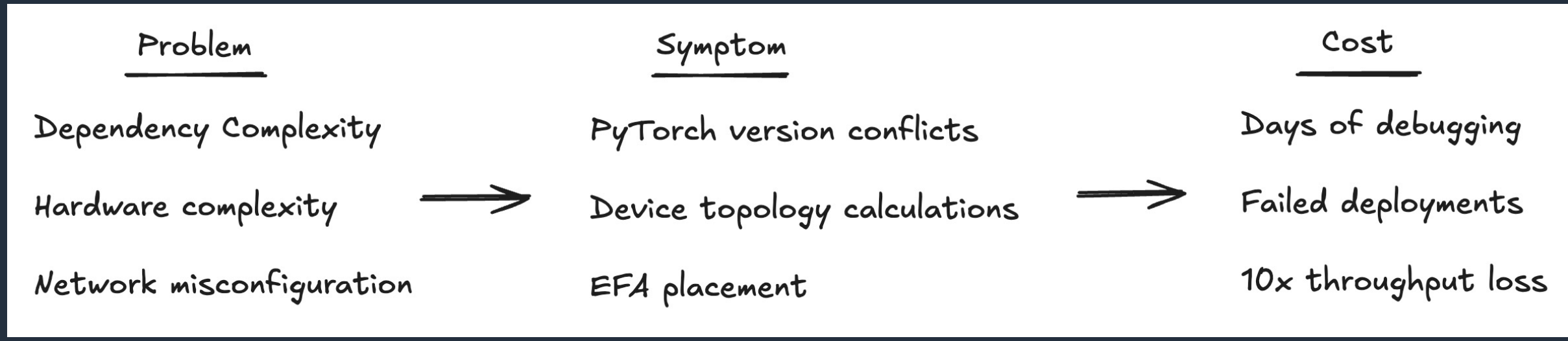
Yahav Biran (yahavb@amazon.com);

Pytorch 26, Paris

Distributed AI Without the Infrastructure Tax

- Deep Learning Containers (DLC) for Build-Time
- Neuron Dynamic Resource Allocation (DRA) for Deploy-Time
- Elastic Fabric Adapter (EFA) DRA for Scale

Why Distributed AI Workloads Are Challenging in Production

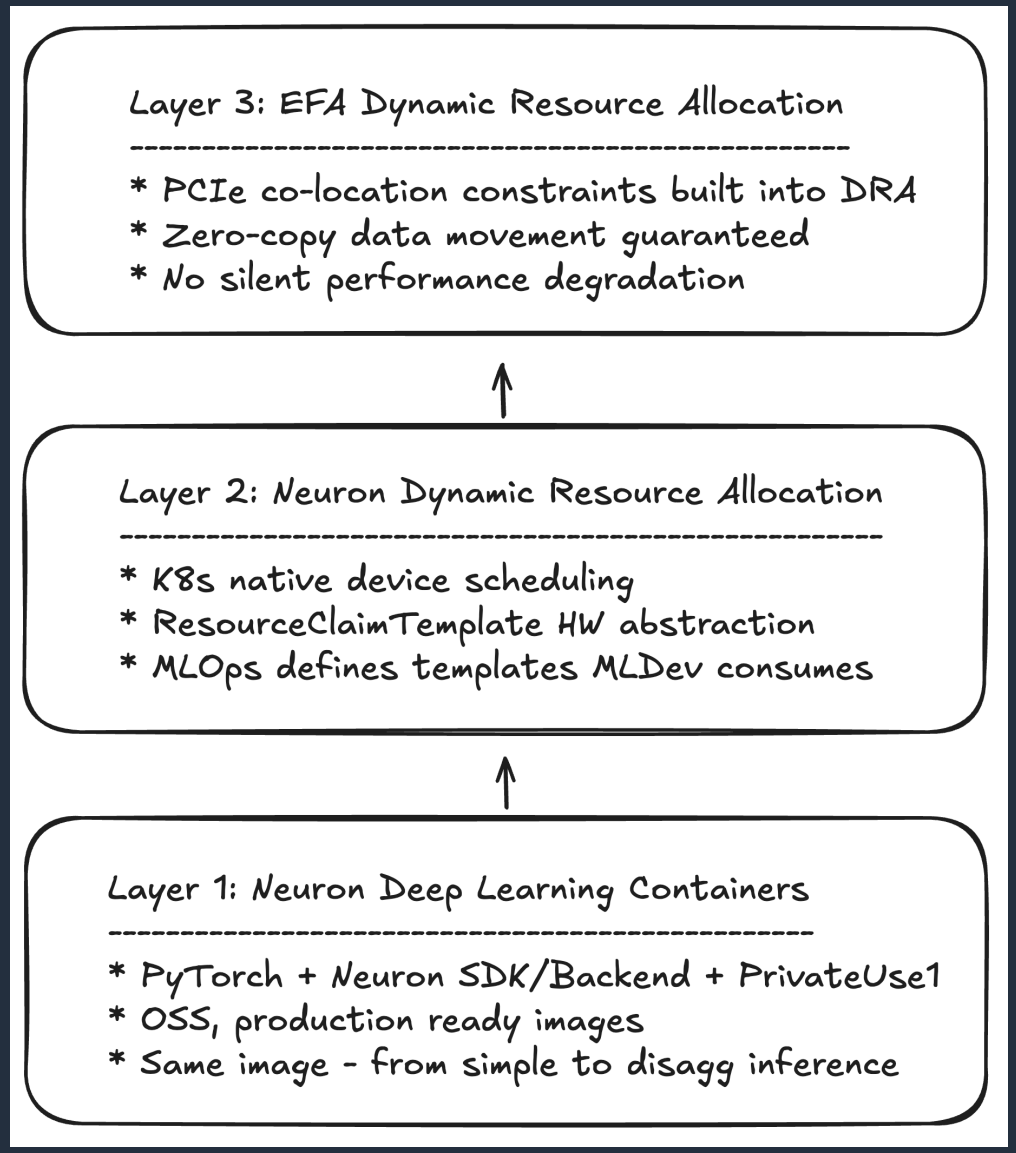


Each problem requires different expertise

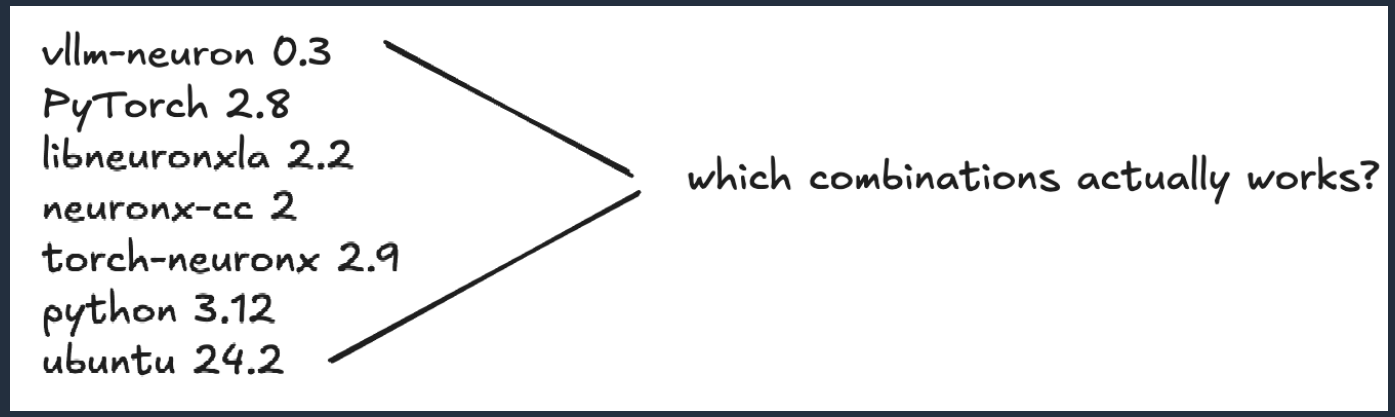
Traditional approach: ML practitioners become infra experts

Result: Weeks of setup time, low cluster utilization

The Solution: DLCs + Neuron+EFA DRA



Versioned Components as a Tested Contract



docker pull

public.ecr.aws/neuron/pytorch-inference-vllm-neuronx:0.13.0-neuronx-py312-sdk2.28.0-ubuntu24.04

- vllm-neuronx — vLLM with Neuron backend
- PyTorch
- neuronx_distributed, neuronx_distributed_inference — Neuron Backend
- torch-neuronx, libneuronxla
- aws-neuronx-tools, nki — Profiling and custom kernels
- libfabric, EFA drivers, Python — OS and drivers

Layer 2: DRA Abstracts Hardware Complexity

K8s-Native Device Scheduling

Before DRA -

ML devs must understand infra

```yaml
schedulerName: my-scheduler  # Custom scheduler required
affinity:
  nodeAffinity:
    requiredDuringSchedulingIgnoredDuringExecution:
      nodeSelectorTerms:
        - matchExpressions:
            - key: workload
              operator: In
              values:
                - mymodel-exclusive
            - key: accelerator
              operator: In
              values:
                - trn2
                - trn3
  podAntiAffinity:  # Ensure only one such pod per node
    requiredDuringSchedulingIgnoredDuringExecution:
      - labelSelector:
          matchExpressions:
            - key: app
              operator: In
              values:
                - llm-inference
        topologyKey: "kubernetes.io/hostname"
tolerations:
  - key: "dedicated"
    operator: "Equal"
    value: "mymodel-trn"
    effect: "NoSchedule"
containers:
  - name: app
    image: my-image
    resources:
      limits:
        aws.amazon.com/neuron: 16  # Magic number - what if instance type changes?
```

Layer 2: DRA Abstracts Hardware Complexity

After DRA — MLOps Defines

```
apiVersion: resource.k8s.io/v1beta1
kind: ResourceClaimTemplate
metadata: { name: l-trn2 }
spec:
  spec:
    devices:
      constraints:
      - { matchAttribute: resource.aws.com/devicegroup4_id, requests: [neurons] }
      requests:
      - name: neurons
        deviceClassName: neuron.aws.com
        allocationMode: ExactCount
        count: 4
        selectors:
        - cel: { expression: "device.attributes['neuron.aws.com'].instanceType == 'trn2.48xlarge'" }
      config:
      - requests: [neurons]
        opaque:
          driver: neuron.aws.com
          parameters: { apiVersion: neuron.aws.com/v1alphav1, kind: NeuronConfig, logicalNeuronCore: 2 }
```

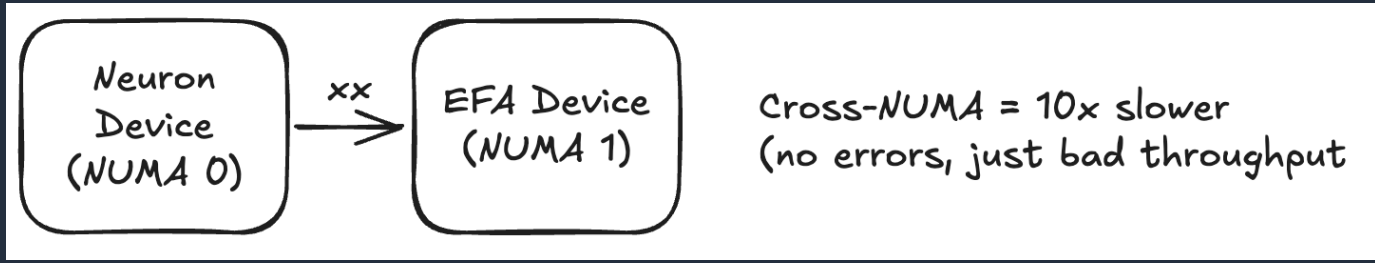
Layer 2: DRA Abstracts Hardware Complexity

After DRA — ML Eng. uses

```
spec:
     resourceClaims:
       - name: l-trn2
         resourceClaimTemplateName: l-trn2
     containers:
       - name: app
         image: public.ecr.aws/neuron/pytorch-inference-vllm-neuronx:0.13.0-neuronx-py312-sdk2.28.0-ubuntu24.04
         command:
           - /bin/bash
         resources:
           claims:
           - name: l-trn2
```

Layer 3: Neuron+EFA DRA

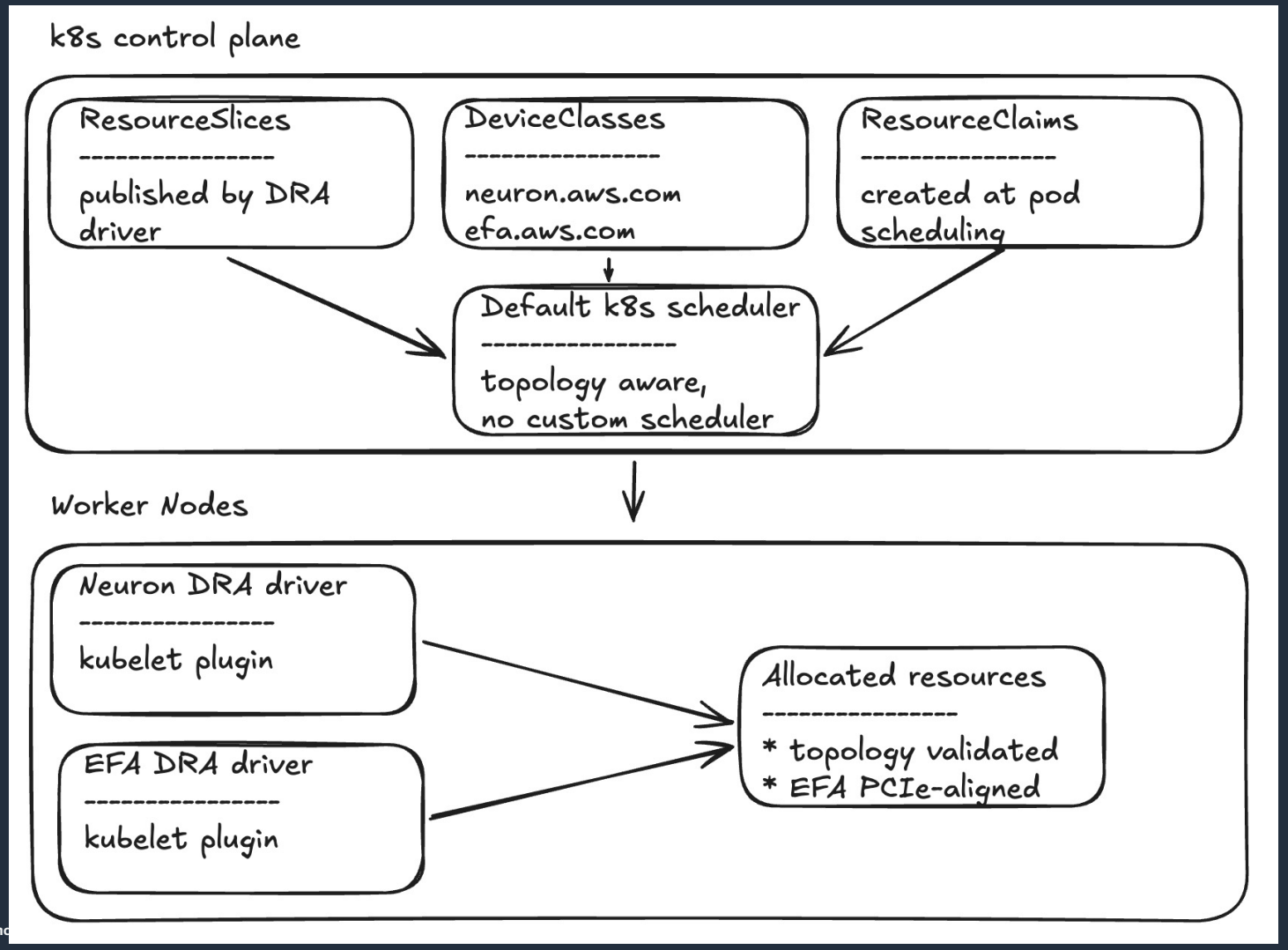
The Silent Performance Problem:



DRA Solution: PCIe Co-Location Constraint

```
apiVersion: resource.k8s.io/v1
kind: ResourceClaimTemplate
metadata:
  name: 1-efa-trn2
spec:
  spec:
    devices:
      requests:
      - name: 4-neurons
        exactly:
          deviceClassName: neuron.aws.com
          count: 4
      - name: 4-efas
         exactly:
           deviceClassName: efa.networking.k8s.aws
           count: 4
       constraints:
       - requests: ["4-neurons", "4-efas"]
          matchAttribute: "resource.aws.com/devicegroup4_id"
```

DRA Architecture: How It Works



Key Takeaways

Deploy Confidently, Scale Efficiently

DLCs as Tested Contracts

https://github.com/aws-neuron/deep-learning-containers

DRA as Hardware Abstraction

- MLOps defines templates with policies
- MLDev reference t-shirt sizes
- Kubernetes-native, no custom schedulers

EFA as Zero-Config Networking

- PCIe alignment validated automatically
- Multi-node topology handled by DRA