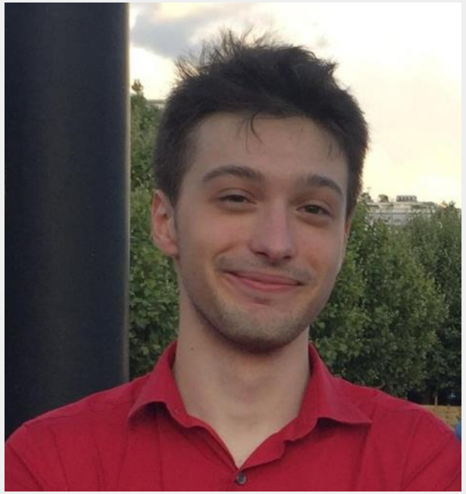
Kubecon US
Scaling AI Inference with Kubernetes and GPUs

NVIDIA

Speaker, Date

Who We Are

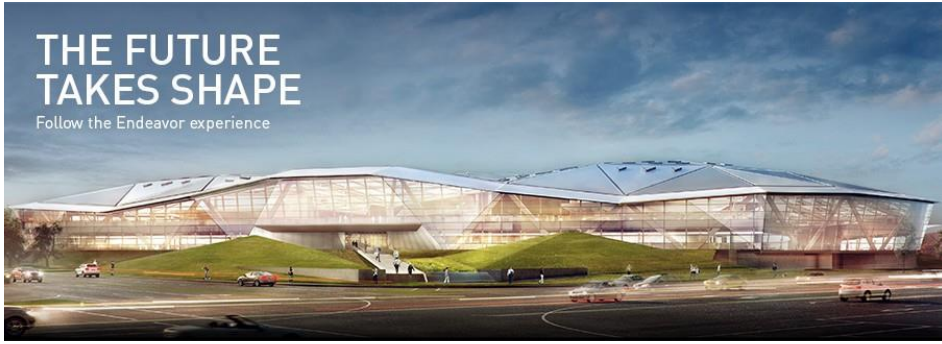


Renaud Gaubert
Containers, K8s & OSS
Mr. Kubernetes

Ryan Olson
DL, HPC & Cloud
Solution Architect

Involvement in the Community

- **February 2017:** Involvement in the community discussions
- **Spring 2017**: Face 2 Face meeting @ Google
- **Summer 2017**: GPUs in K8s Design doc
- **Kubernetes 1.8**: Alpha Feature available
- **Kubernetes 1.10:** Beta Feature available
- **Spring 2018**: Face 2 Face meeting @ NVIDIA
- **Kubernetes 1.12:** GPU Monitoring in K8s
- **Kubernetes 1.13:** Alpha GPU Monitoring



AGENDA

Scaling AI Inference with Kubernetes and GPUs

Why do we care?

Scaling with *GPUs*

AI Inference Pipeline

Scaling with *Kubernetes*

Why Do We Care?

AI Inference Is Exploding

Creating a $20 Billion Opportunity in Next 5 Years

1 Billion

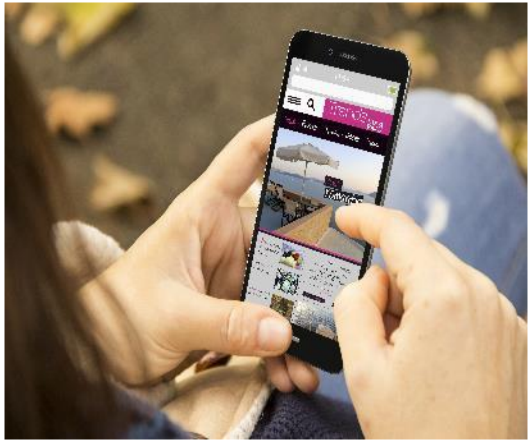
Videos Watched Per Day
Facebook

LIVE VIDEO

1 Billion

Voice Searches Per Day
Google, Bing, etc.

SPEECH



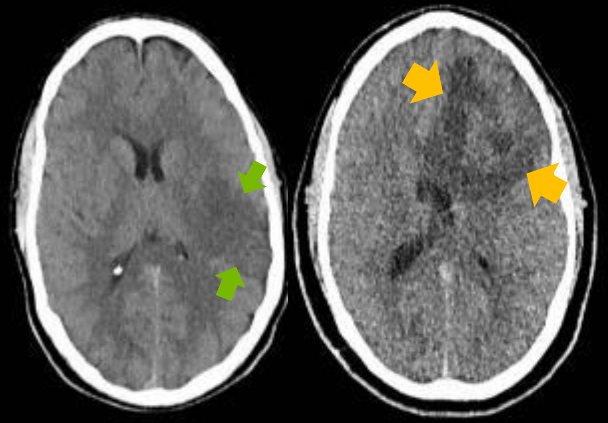
1 Trillion

Ads/Rankings Per Day
Impressions

RECOMMENDATIONS

AI Transforming Every Industry

HEALTHCARE



>80% Accuracy & Immediate Alert to Radiologists

INFRASTRUCTURE

50% Reduction in Emergency Road Repair Costs

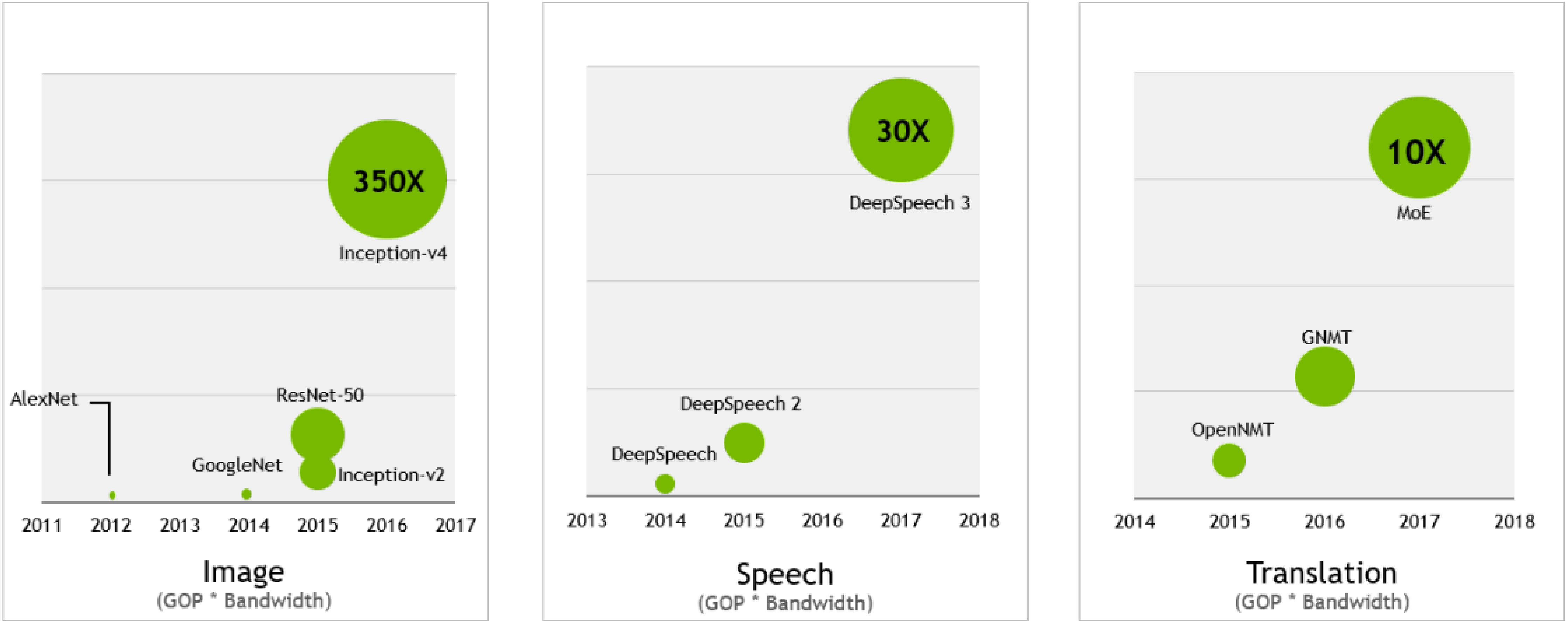
IOT

>$6M / Year Savings and Reduced Risk of Outage

Scaling with GPUs

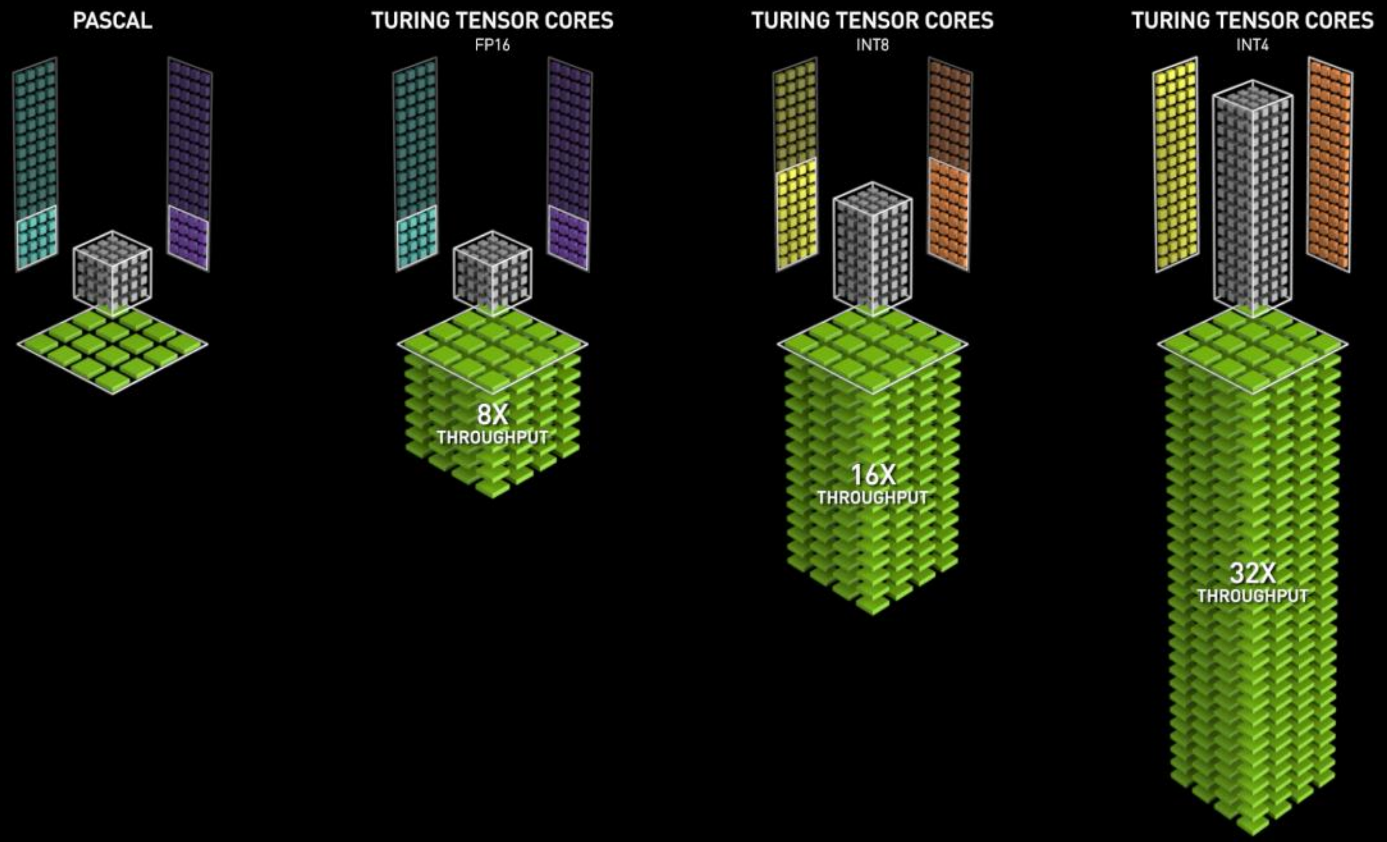
Neural Network Complexity Is Exploding

Bigger and More Compute Intensive



NEW TURING TENSOR CORE

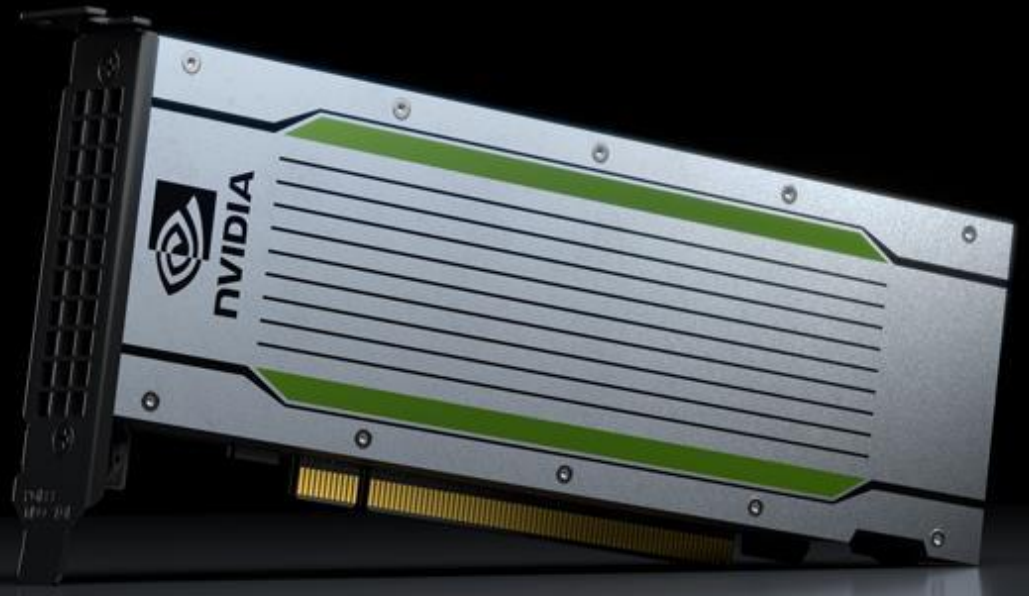
MULTI-PRECISION FOR AI INFERENCE
65 TFLOPS FP16 | 130 TeraOPS INT8 | 260 TeraOPS INT4

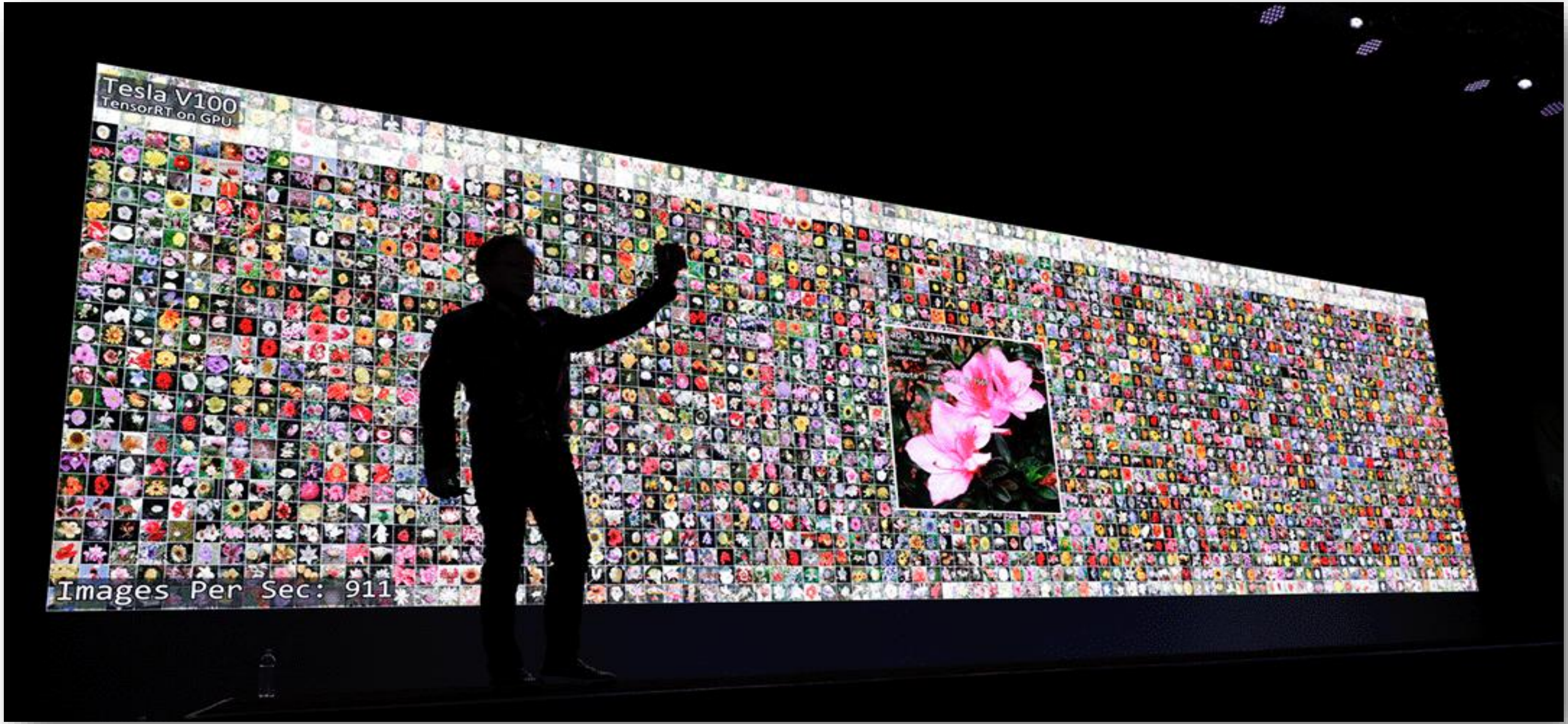


TESLA T4

WORLD'S MOST ADVANCED INFERENCE GPU

Universal Inference Acceleration
320 Turing Tensor cores
2,560 CUDA cores
65 FP16 TFLOPS | 130 INT8 TOPS | 260 INT4 TOPS
16GB | 320GB/s




Tesla V100
TensorRT on GPU
Images Per Sec: 911

AI Inference Pipeline

Compute Pipeline

1. Input Data from Source
2. Transform Input → Input Tensors (on CPU or GPU)
3. Input Tensors → GPU memory
4. Compute
5. Output Tensors → Host memory
6. Transform Output Tensors → consumable Output value

- BEST Performance / Value = Keeping the Pipeline FULL
- Integrating HPC best practices into data center workloads

Where are the Bottlenecks?

- Ingest
  - Moving Input to Compute (gb/sec)

- Input → Input Tensors (reversed for Output)
  - What is the compression ratio for common problems?
  - Computational Time to Transform?

- Ratio of Compute vs. Transfers
  - Goal: Evaluation of the DNN is the rate limiting condition

- **Success** = Proper choice of Hardware, Software and Tuning Parameters

Compute → Pre/Post → Serving → Metrics → Kubernetes

Inference Compute Options

TensorRT

Performance
Memory Footprint
Control over Precision (fp/int)
Deployable Package
Lowest DNN Compatibility

Framework + TRT

Framework Fallback for
Unsupported TRT Layers
Framework Overheads
Allocation Ownership Issues

Framework

Most DNN Compatibility
Most Overhead
Least Performant

Preferred ←

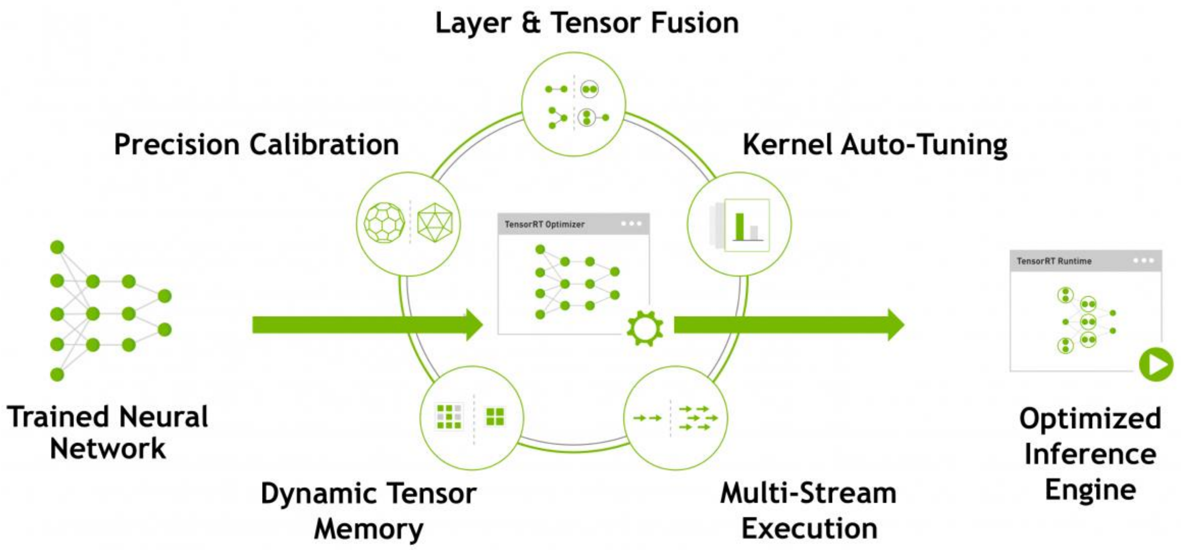
TensorRT

Designed to deliver maximum throughput and efficiency

Runs in two phases: build and deployment

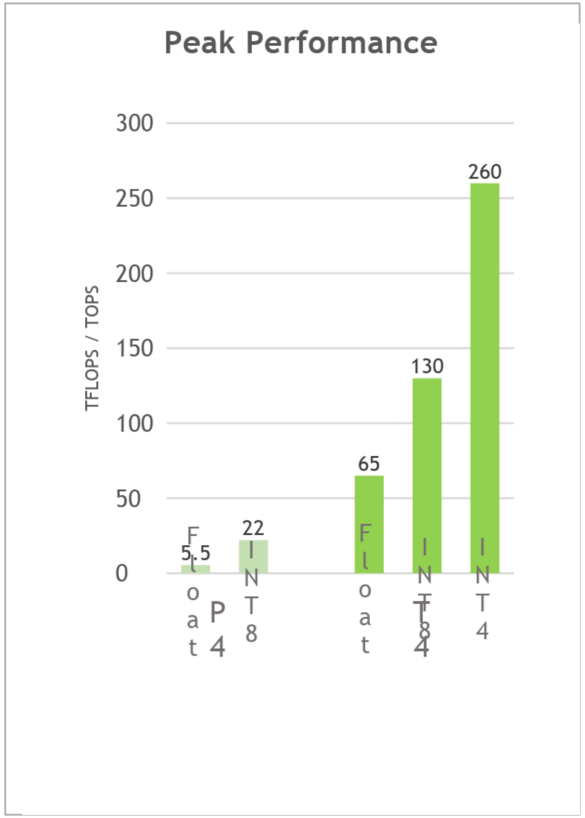
The build phase optimizes the network for target hardware and serializes result

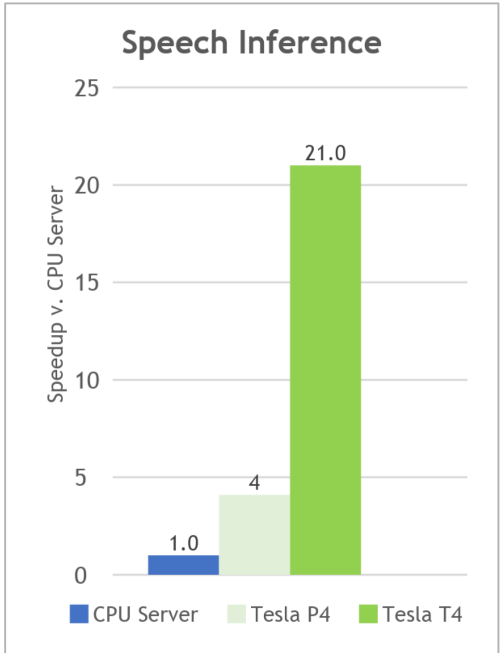
Deployment phase executes on batches of input without any deep learning framework



WORLD'S MOST PERFORMANT INFERENCE PLATFORM

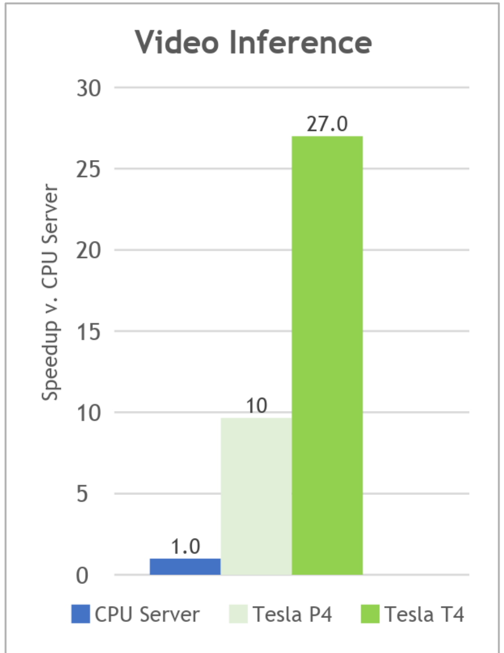
Up To 36X Faster Than CPUs | Accelerates All AI Workloads





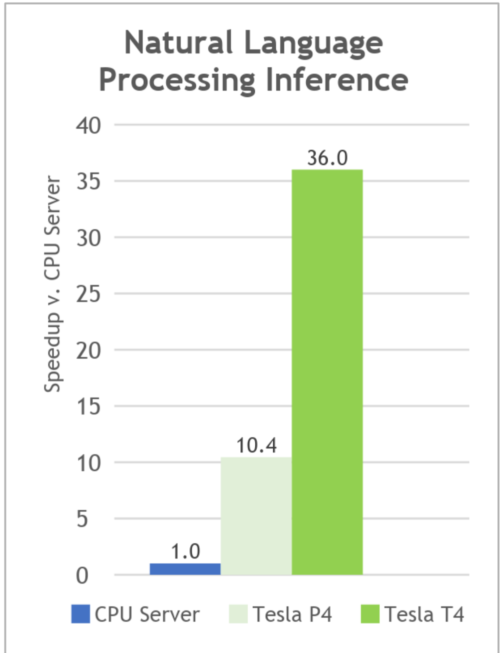
Speedup: 21X faster

DeepSpeech 2



Speedup: 27x faster

ResNet-50 (7ms latency limit)



Speedup: 36x faster

GNMT

Compute → <u>Pre/Post</u> → Serving → Metrics → Kubernetes

Pre/Post Processing

- Problem Specific
- Requires the same level of attention as evaluating the DNN compute
- Questions
  - CPU vs. GPU (video decode example)
  - Location
    - IN-Process (same memory space)
    - IN-Pod (shared IPC spaces, i.e shared memory, /tmp
    - IN-Node (co-located on the same node via Pod Affinities)
      - May need hacks to break down namespace barriers
      - Scaled independently
    - Fully Independent
- Answer: Data Movement is Key

Coupled / scaled jointly

Compute → Pre/Post → <u>Serving</u> → Metrics → Kubernetes

NVIDIA TensorRT INFERENCE SERVER

Containerized Microservice for Data Center Inference

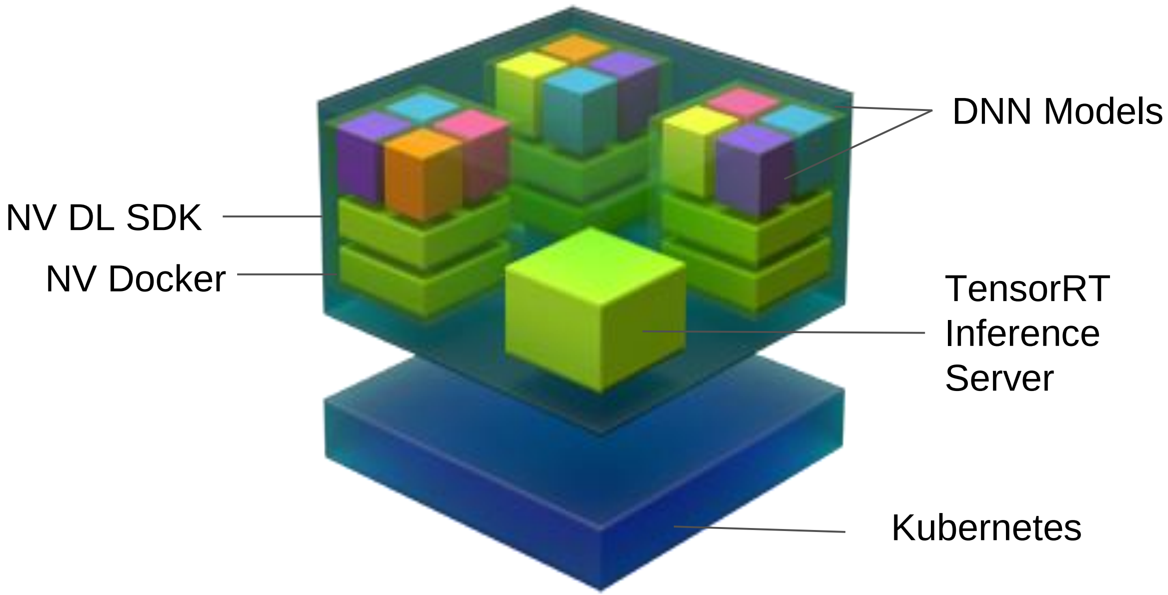
Tunable Concurrency

Multiple models scalable across GPUs

Supports all popular AI frameworks

Seamless integration into DevOps deployments leveraging Docker and Kubernetes

Ready-to-run container, free from the NGC container registry



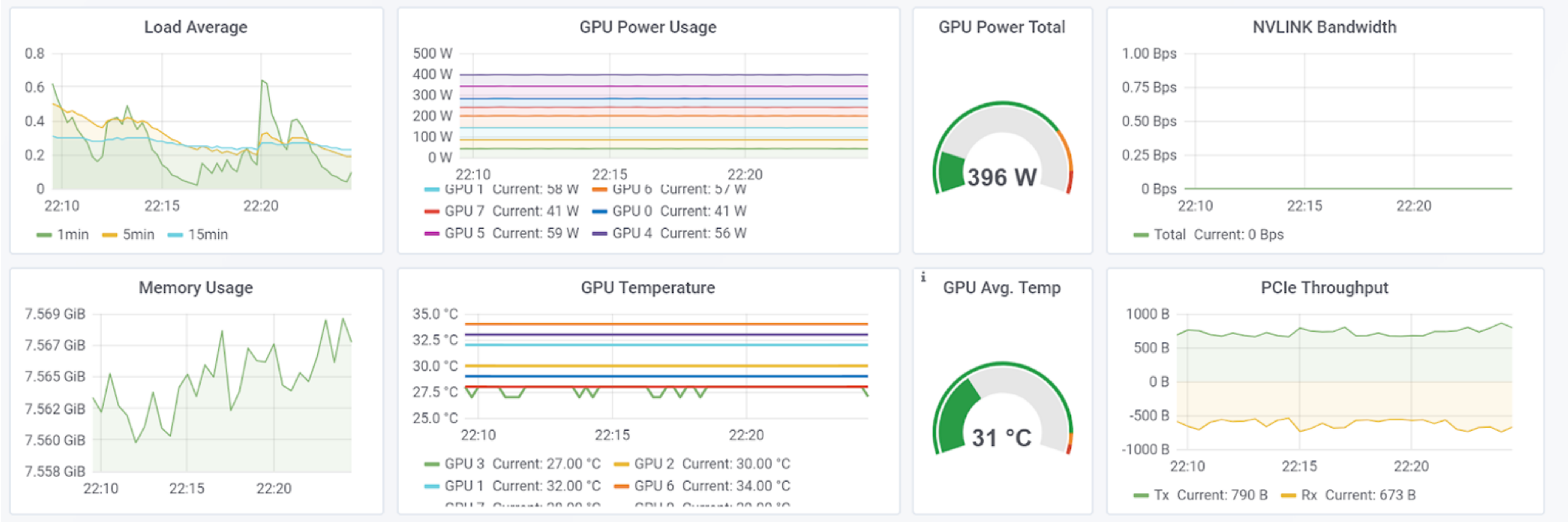
Compute → Pre/Post → Serving → <u>Metrics</u> → Kubernetes

AVAILABLE METRICS

Category	Name	Use Case	Granularity	Frequency
Utilization	Power usage	Proxy for load on the GPU	Per GPU	Per second
	Power limit	Maximum GPU power limit	Per GPU	Per second
	GPU Utilization	GPU utilization rate [0.0 - 1.0)	Per GPU	Per second
Count GPU & CPU	Request count	Number of inference requests	Per model	Per request
	Execution count	Number of model inference executions Request count / Execution count = Avg dynamic request batching	Per model	Per request
	Inference count	Number of inferences performed (one request counts as “batch size” inferences)	Per model	Per request
Latency GPU & CPU	Latency: request time	End-to-end inference request handling time	Per model	Per request
	Latency: compute time	Time a request spends executing the inference model (in the appropriate framework)	Per model	Per request
	Latency: queue time	Time a request spends waiting in the queue before being executed	Per model	Per request

Monitoring

```
$ helm repo add nvidia https://nvidia.github.io/gpu-monitoring-tools/helm-charts
$ helm install nvidia/prometheus-operator
$ helm install nvidia/kube-prometheus
```

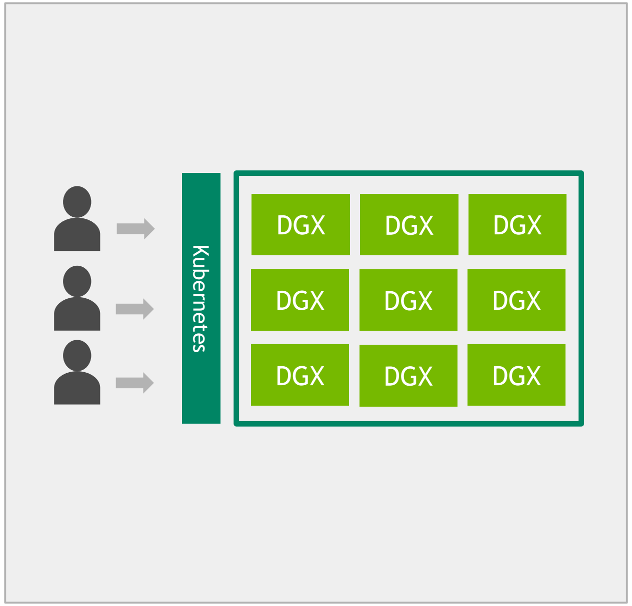


Compute → Pre/Post → Serving → Metrics → Kubernetes

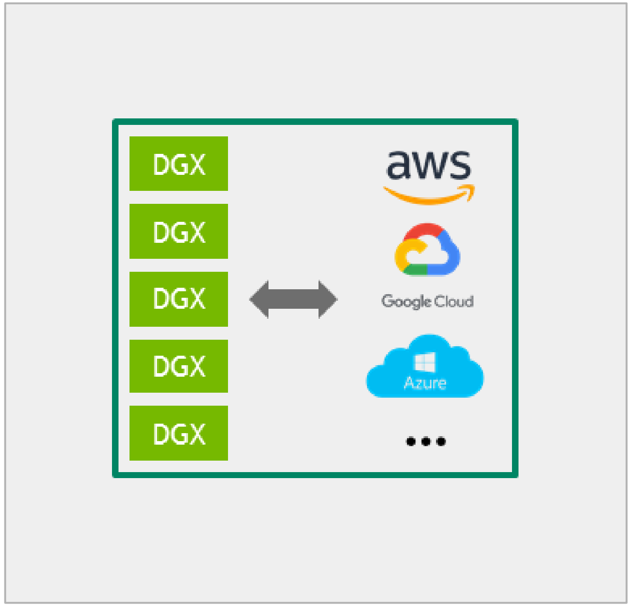
Scaling With Kubernetes

Kubernetes

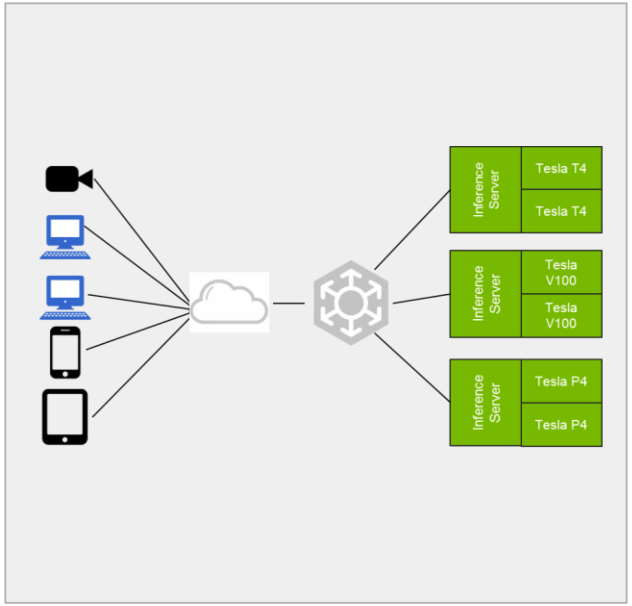
Use cases for GPU Powered Applications



Resource Attribution
Many users, many nodes

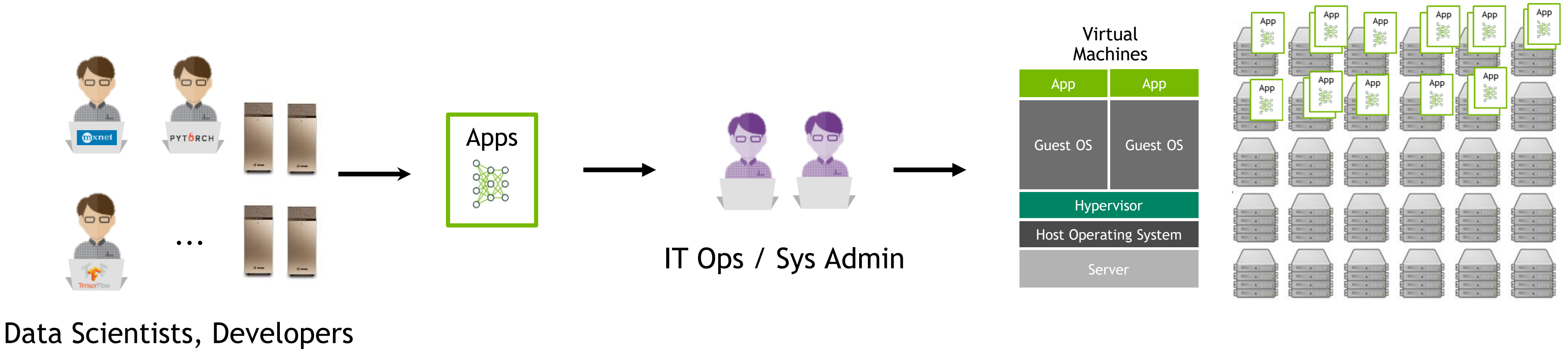


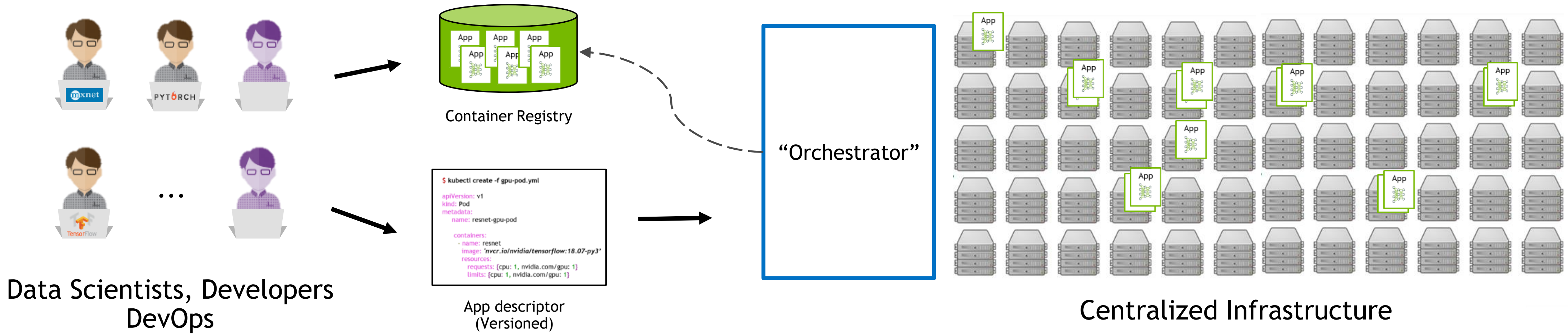
Cloud bursting



Production Inferencing

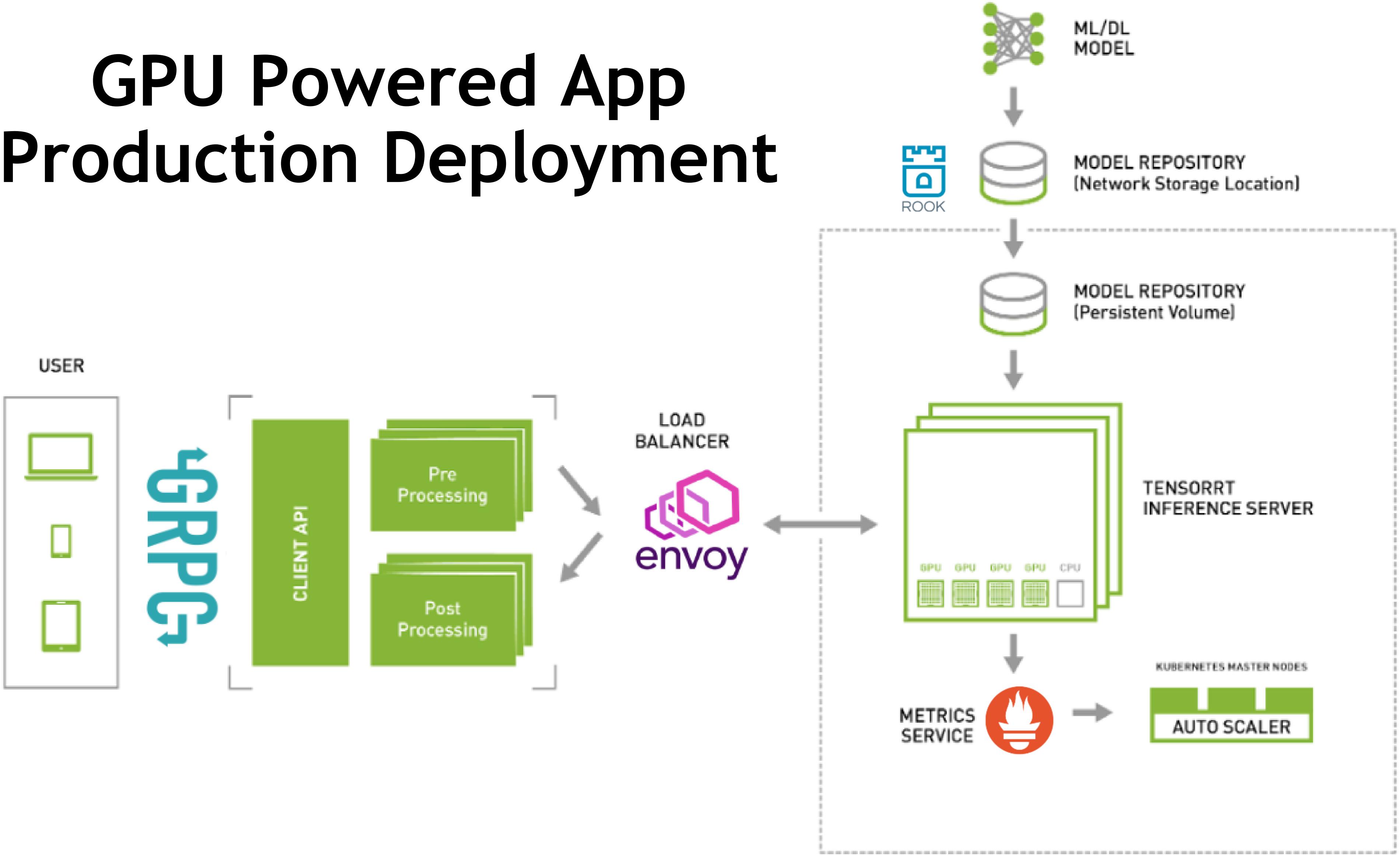
AI DEPLOYMENTS - THEN AND NOW





How does it fit with K8s?

GPU Powered App Production Deployment



Pitfalls of Kubernetes

Inside Kubernetes Resource Management (Kubecon EU18)

Class	CPU	Memory
Best Effort Requests = Limits = 0 (all Containers)	R(equests) L(imits)	R(equests) L(imits)
Burstable 0 < Requests <= Limits (at least one Container)	R L	R L
Guaranteed 0 < Requests = Limits (all Container)	R L	R L

K8s uses CFS Quotas to enforce CPU limits. There is a known bug affecting well behaved applications by CPU throttling them.

End of Talk

Speaker, Date