Foundation Models Stack

Foundation Models Stack
Cloud-native stack for on-prem and multi-cloud use

- **5a** NLP
- **5b** Code
- **5c** Chemistry & Materials
- **5d** Sensor
- **5e** Cybersecurity
- **5f** Geospatial

4 **Foundation Model UX and Toolchain**
Front-end, orchestration, multi-cloud support
pre-built workflows, evaluation metrics, testing & validation, etc.
CodeFlare
Training
Fine tuning
Inference

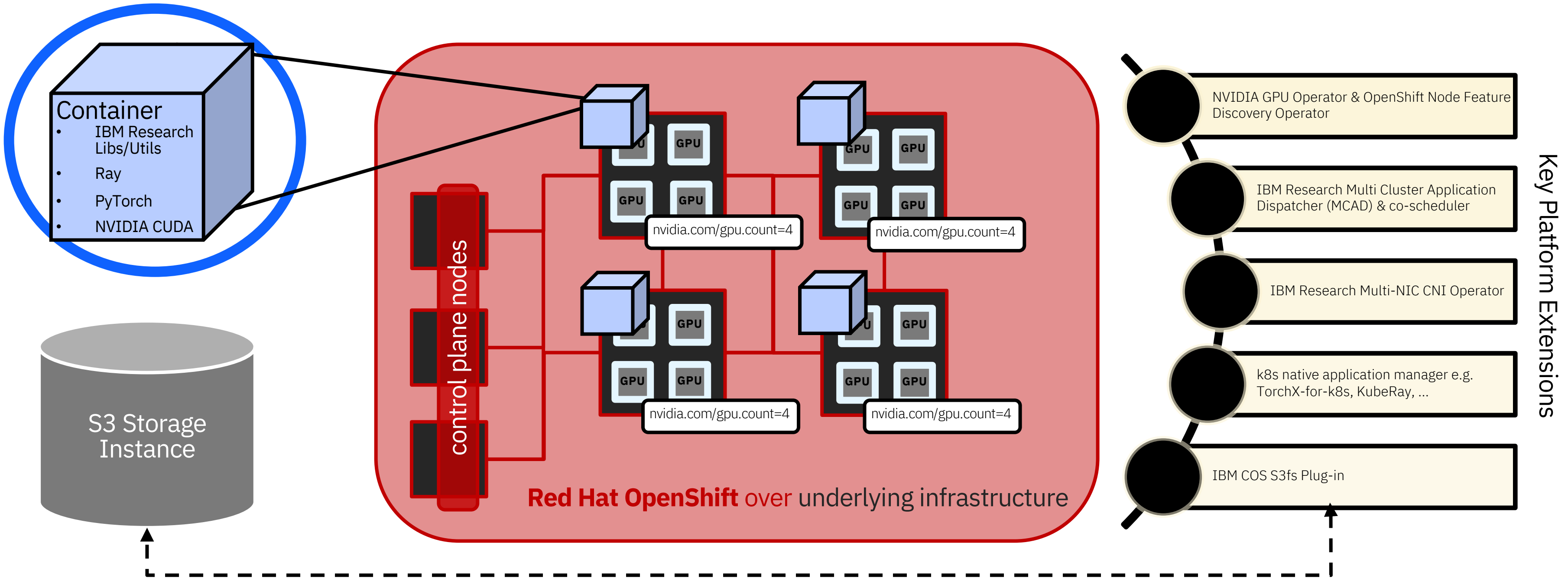
3 **Scale-out Middleware for Training and Inference**
Workflows/pipelines, Ray and PyTorch
RAY
PyTorch

2 **Hybrid Cloud Platform for Training and Inferencing**
Distributed resource management, scheduling
OpenShift scale, high performance networking, CNI/GPU enablement
Red Hat OpenShift

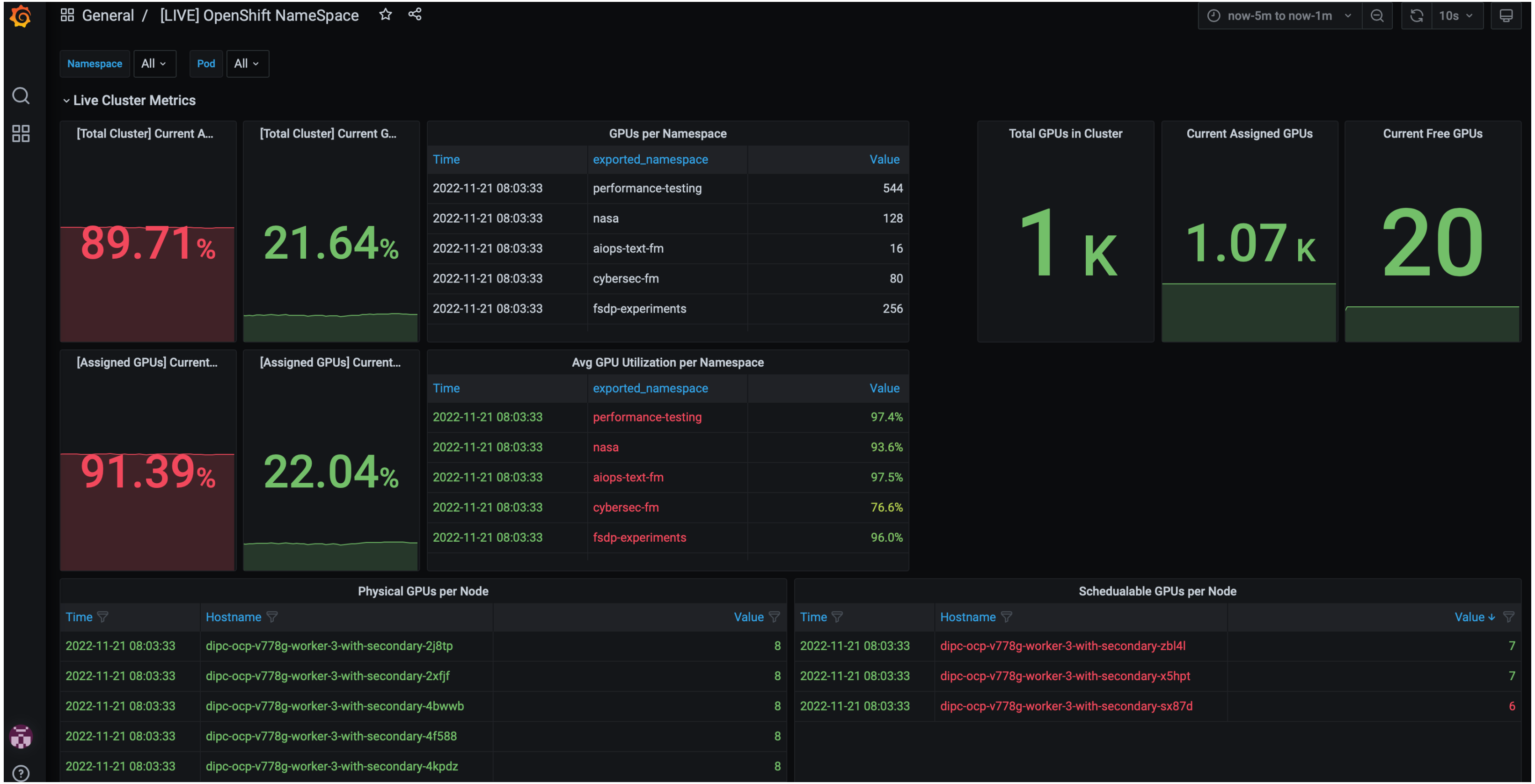
Scale-out Infrastructure
1a GPU Environment in IBM Cloud
Other public clouds
AI accelerators (e.g. Sentient) **1b**

FM stack: Training View

A Bird's Eye View of the Hybrid Cloud Layers



FM Stack Observability and Automation



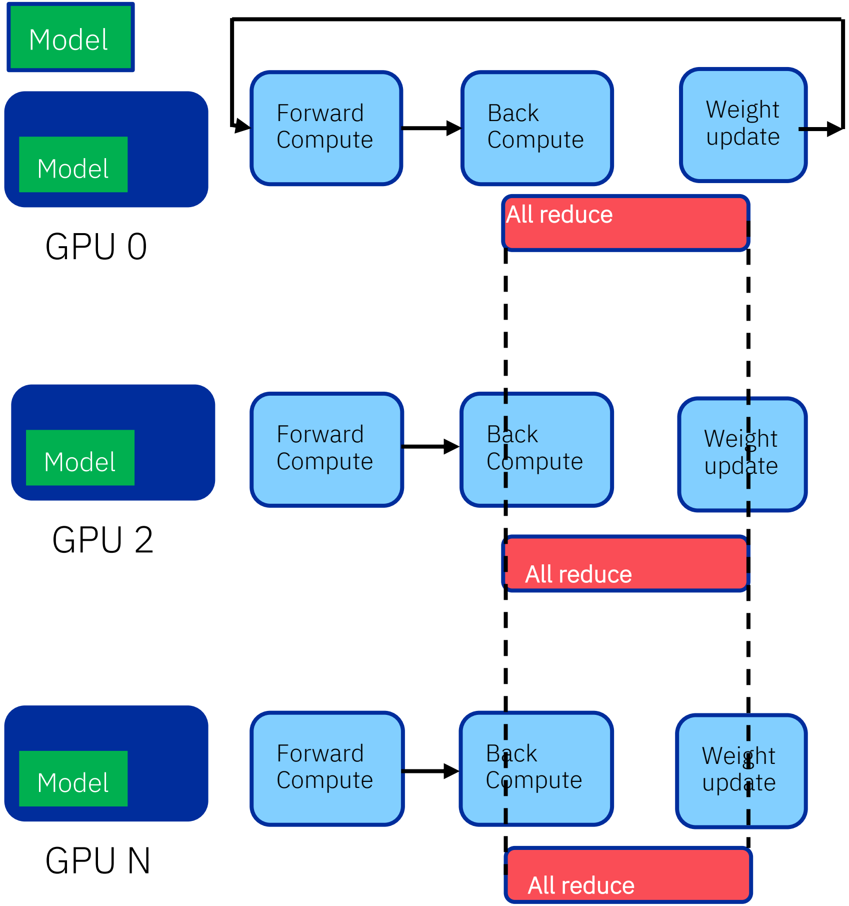
Grafana dashboard; OpenShift console (internal links only)

FM Training: Recent Improvements

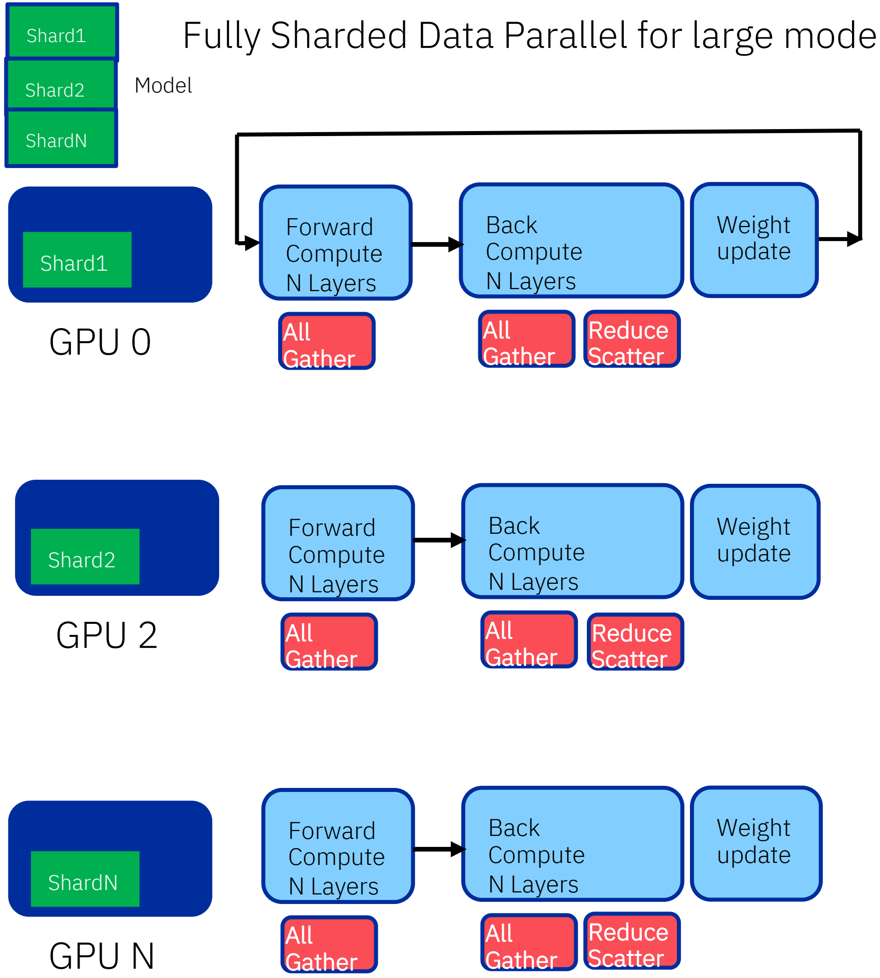
- Primary areas of engagement with PyTorch
  - Fully Sharded Data Parallel (FSDP) for training >1B parameter models
  - Seamless launching of jobs from laptop using TorchX

- Key highlights
  - Changes to FSDP APIs to scale on Ethernet 5x more efficiently
  - Talk at PyTorch conference (Dec 2nd) with joint blog published
  - IBM research blog

Distributed Data Parallel for Models that fit in a GPU



Fully Sharded Data Parallel for large models



Computation vs Communication

Model training parameters:

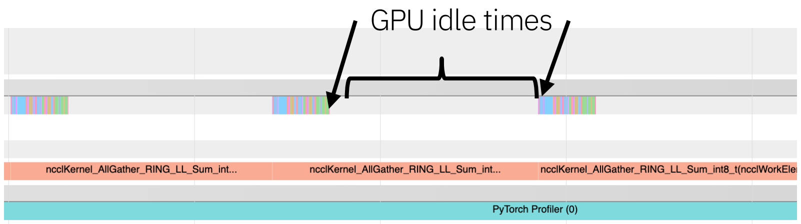
- s – sequence length
- h – hidden state (embedding) size
- B – batch size
- V – vocabulary size

- T5 (11B) compute: $96 \times (54Bsh^2 + 12Bs^2h) + 6BshV$ (dominated by first term)

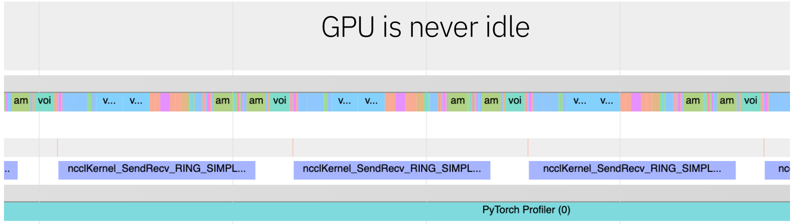
- T5 (11B) communication: $24 \times 27h^2$

- Compute to communication ratio: $4 \times (2Bs)$-dominant term

- To improve compute communication ratio, primary knob is B (s is usually fixed for a class of models)

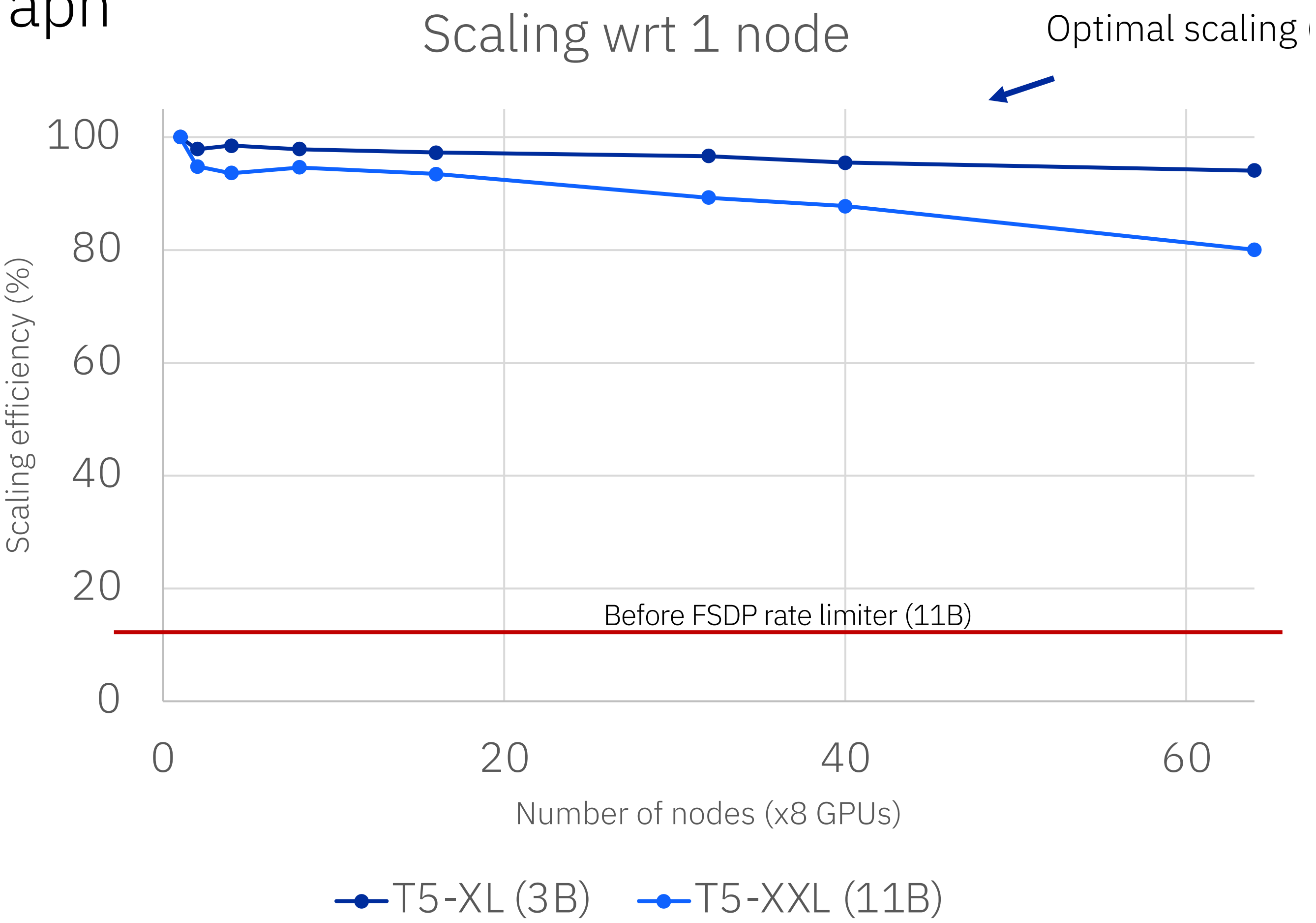


Before

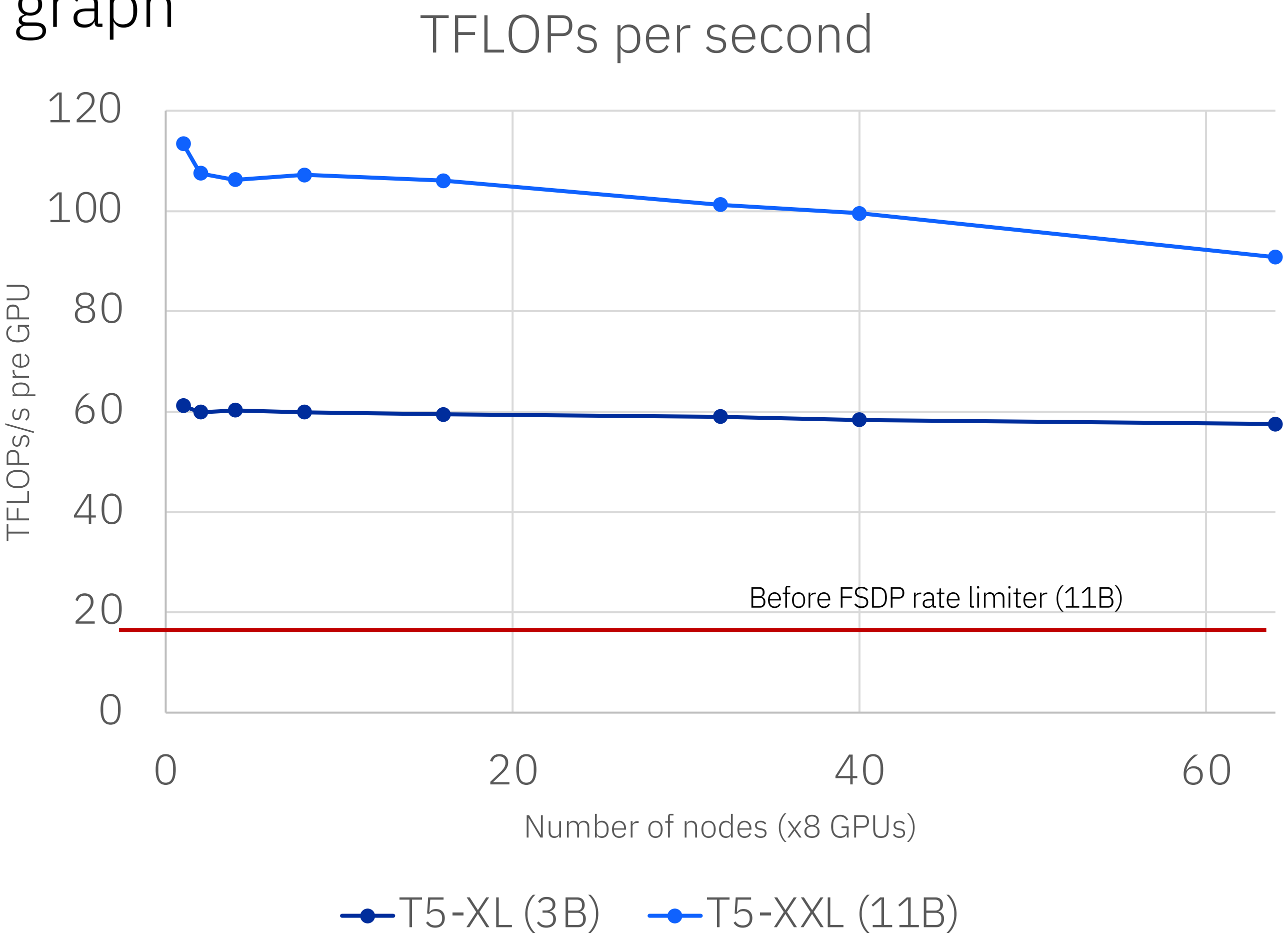


After

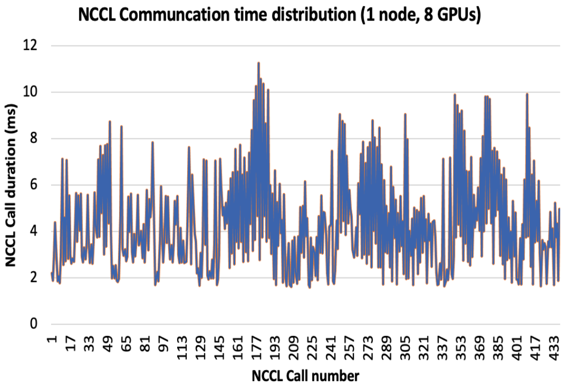
Scaling graph



Teraflops graph



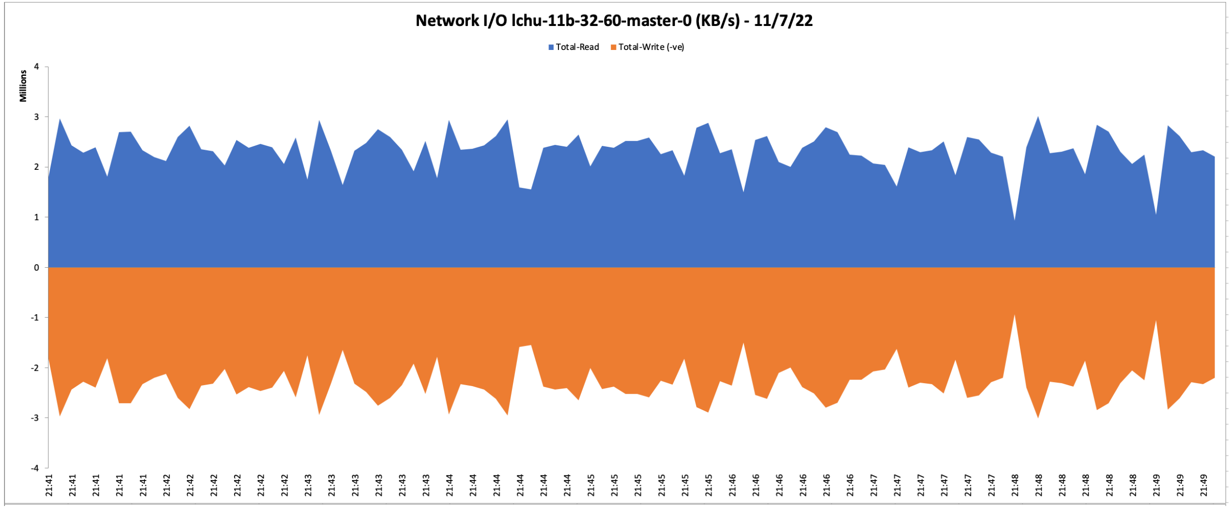
Intranode and Internode bandwidth



Intranode NVLink bandwidth :
~1 – 5.4GB/s
Avg: 3GB/s (1.2%)

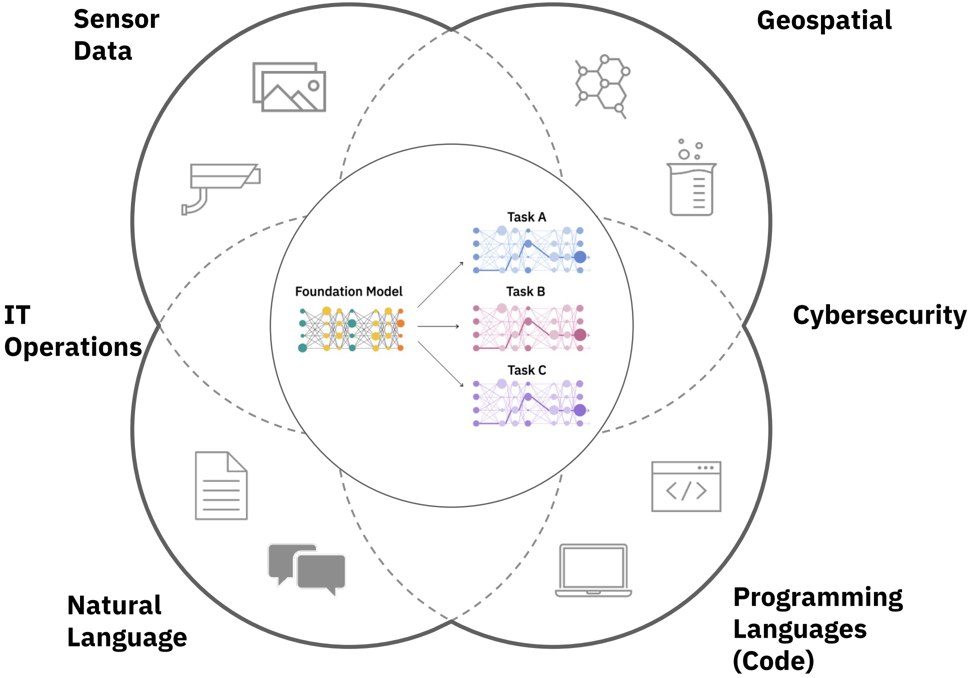
Key lessons:

- For training, latency is important to keep up with GPU computation
- Bandwidth is not a critical bottleneck (at 11B)



Internode Network Bandwidth:
~1 – 4 GB/s
Avg: 2.5 Gbps (10%)

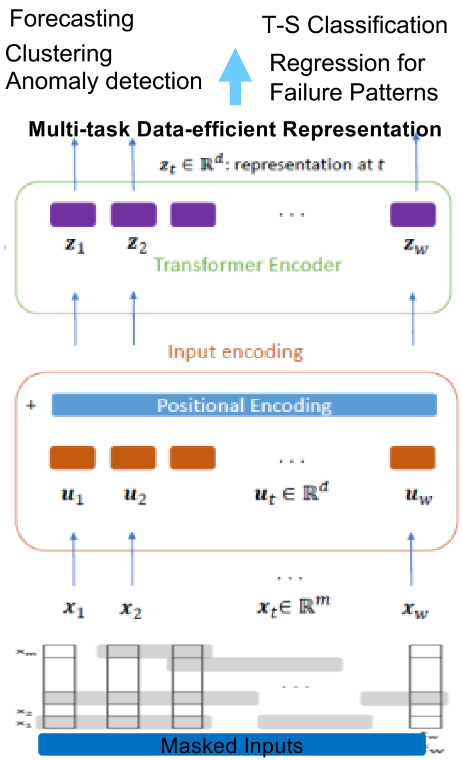
FM Verticals: Recent improvements on non-NLP Models



In many domains, there are large amounts of unlabeled data available in enterprises.

This can used to train foundation models, which can solve business problems that were previously considered intractable.

Timeseries transformers



Time-Series data: $\mathbf{x}_t$

- Attention used to learn temporal signature
- Masking to enable unsupervised training

– Features
  - transformer architectures for time-series, utilities for data transform (sliding windows)
  - Prediction, Forecasting & Classification
  - Pre-training/Fine-tuning
  - Appropriate back-end support to launch data parallel training and tuning with scalability up to 128 GPUs on software stack
  - Later: multivariant/covariant structures

– Stack (for training): available for GCP, AWS, Azure and openShift

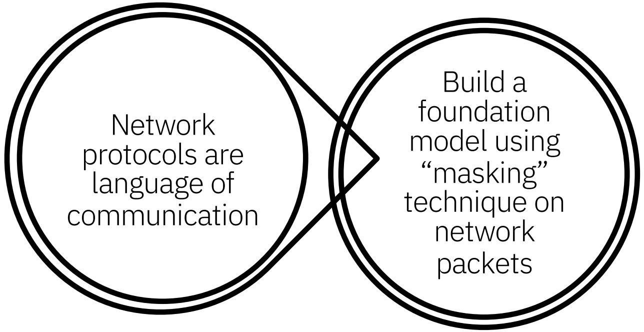
– UX: Jupyter notebook

– Accuracy Improvements: 10-30%

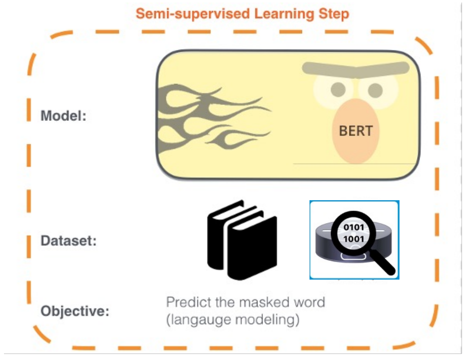
– Error reduction: 40-50%

- LNG Trains (Regression)
  - 100GB; 24M model
  - MSE error: 0.0332 → 0.0175
- Auto DTC Codes (Event Sequence - Prediction)
  - 8GB; 3M model
  - Recall: 0.11 → 0.4;
  - Precision: 0.55 → 0.6; F1: 0.11 → 0.43
- Oil Wells (Classification)
  - 32 GB; 24M model
  - F1 score: 0.68 →0.75
- Chemical Process (Prediction)
  - 100Mb; 1M model
  - R2: 0.65 → 0.78
- Waste-Water treatment (Prediction)
  - 50 MB; 500K parameters
  - MSE: 0.12 → 0.10
- Medical Device Demand forecasting
  - MAPE: 0.5 → 0.19

Network Data Transformers



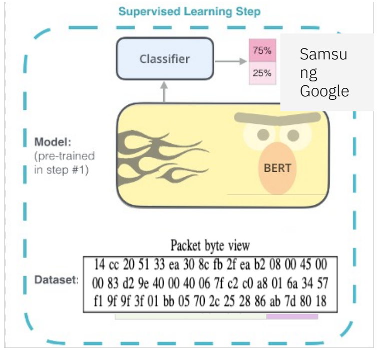
Variable	Description	Value
D	Dimension of embeddings	128
N	Number of successive FQDNs in a sequence	[8,16,32,64]
k	Truncation level in FQDN hierarchy (Sec. III-B)	3
h	Number of BERT self-attention heads	8
b	BERT batch size	32
l	Number of attention layers	4
l_r	Learning rate	0.001
w	Weight decay	0.01



- Construct BPE (byte pair encodings) using knowledge of the network protocol (e.g., 4 octet IP address, 2 octet port numbers)
- Leverage knowledge of network packet encapsulation to build a tree structure over the BPEs (e.g., IP → TCP → HTTP)

Train foundational model on raw packets and down stream task on device classification

Task	Embedding	Training Dataset Testing Subset (Training:Testing split)	Independent Validation Dataset
Device Type	Random	0.997	0.592
	GloVe	0.994	0.585
	Norbert	0.998	**0.965**
Manufacturer	Random	0.996	0.588
	GloVe	0.998	0.726
	Norbert	0.981	**0.906**



20-40% gain on downstream tasks of network packet traces

Foundation Models on Ops Data

Identified based on few qualitative requirements:

- ➢ Sequences whose ambiguity decrease with context width
- ➢ Large volume of unlabeled data
- ➢ Downstream tasks and baseline measures for validation (rules/non-FM AI/ML models)

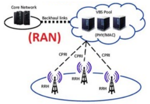
Tech Notes
Customer care, Product docs

- Data: Combination of external data sources (stack overflow, RFCs, product manuals, customer tickets)
- Modality: text
- Downstream tasks: tech Q&A, incidence similarity search

stackoverflow

Metrics, Logs, traces
V/CNFs in Telco Core and RANs

- Data: Observability data from network functions spanning metrics, logs and traces
- Modality: timeseries (numeric and categorical)
- Downstream tasks: continuous optimization, configuration recommendation, fault diagnosis



Cybersecurity
Host/Network IDS

- Data: Host and network telemetry data (process tree, file accesses, network flows)
- Modality: graph (cross event dep)
    - High disk I/O → suspicious
    - Code42 spawns a process that has high disk I/O → legitimate
- Downstream tasks: device quarantine, threat hunting

CROWDSTRIKE

Geospatial Transformers

Goal

Create mind share within the developer community on IBM’s eminence in Foundation Models by democratizing geospatial and weather intelligence.

Initial focus

Enable users to detect environmental changes in order to understand impacts on human and natural systems.

FM for Environmental Change Detection

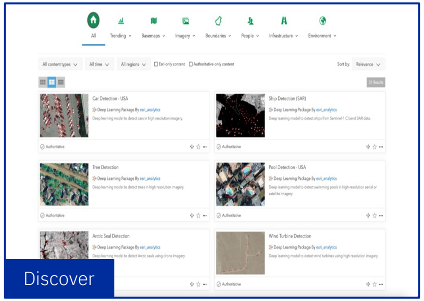
Data set:

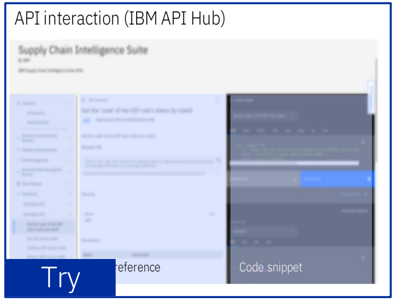
Harmonized Landsat Sentinel-2 & HOD Weather data

New capability for developers:

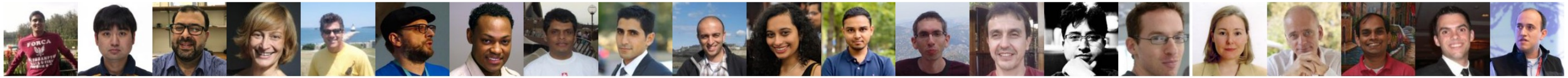
- Ability to combine pretrained models for weather and satellite data
- Focused on change detection using small subset of labeled data that user provides
- Developer then uses the model for detection and monitoring purposes in their application.

Examples:
Ground asset monitoring & risk analytics
Weather and environmental impact monitoring
Land use changes – biomass, agriculture
Disaster response and management





Foundation models team



Abhishek Malvankar · Akihiro Kishimoto · Alaa Youssef · Aleksandra (Saska) Mojsilovic · Alex Mastri · Alexy Khrabrov · Ali Sydney · Alim Alim · Alon Halfon · Apo Kayi · Apoorva Nitsure · Apoorve Mohan · Ariel Gera · Artem Spector · Avi Sil · Avishai Gretz · Bengi Karacali-Akyamac · Bernard Metzler · Bishwaranjan Bhattacharjee · Brian Belgodere · Carlos Andrade Costa

Cezar Pendus · Chulaka Gunasekara · Claudia Misale · Constantinos Evangelinos Constantinos Evangelinos · Dakshi Agrawal · Danish Contractor · Dave Singer · David Kung · Davis Wertheimer · Dennis Wei · Diana Arroyo · Dong Chen · Donny Rose · Doru Bercea · Edward Barth · Edward Epstein · Elham Khabiri · Erik Altman · Errol Ruschil · Gift Sinthong · Gil Shurek

Graeme Blackwood · Gustavo Tellez · Hao Yu · Hsianghan Hsu · I-Hsin Chung · Igor Melnyk · Ilya Shnayderman · Inkit Padhi · Issei Yoshida · Jamie Yang · Jayant Kalagnanam · Jeffrey Burns · Jerret Ross Jerret Ross · Jian-Ming Xu Jian-Ming Xu · John R. Smith · Jose Brunheroto · Juergen Bross · Karthikeyan Natesan Ramamurthy · Kun-Lung Wu · Leland Chang · Lena Dankin

Liat Ein-Dor Liat Ein-Dor · Liling Li · Linsong Chu · Liran Schour · Lisa Hamada · Lixiang Luo · Luis Lastras · Marc Dombrowa · Marina Danilevsky · Marquita Ellis · Masayasu Muraoka · Mattia Rigotti · Maurizio Drocco · Michele Merler · Ming-Hung Chen · Mohamed Ghalwash · Mudhakar Srivatsa · Nam Nguyen · Noam Slonim · Olivier Tardieu · Orith Toledo-Ronen

Paul G Crumley · Pavithra Ranganathan · Pavlos Maniotis · Payel Das · Pedro Bello-Maldonado · Pierre Dognin · Priya Krishnan · Priya Nagpurkar · Radu Florian · Raghu Kiran Ganti · Rama Divakaruni · Rameswar Panda · Richard Allen Young · Rishabh Garg · Robert Haas · Robert Walkup · Rudra Murthy V · Sachin Joshi · Salim Roukos · Saswata Adhikari · Seetharami Seelam

Seiji Takeda · Sharon Keidar-Barner · Shila Ofek-Koifman · Shyam Ramji · Siva Sankalp Patel · Sophia Wen · Sunil Shukla · Sunyanan Choochotkaew · Supriyo Chakraborty · Takeshi Yoshimura · Talia Gershon · Tatsuhiro Chiba · Tejas Indulal Dhamecha · Tetsuya Nasukawa · Todd Takken · Todd Ward Todd Ward · Tonia Elengikal · Viji Srinivasan · Yair Schiff · Yang Zhao · Yoav Katz

Yoonho Park · Young-Suk Lee · Yousef El-Kurdi · Youssef Mroueh · Yuan Chi Chang

IBM

OSS Mindshare

Middleware and platform stack intersecting key open-source projects

- **Toolkits** integrating common tools and streamlining user experience, delivered as part of **CodeFlare** project
- Enhancing **PyTorch** stack integration with OpenShift and Ray for scalable and simplified distributed training on OpenShift
- Extending **Ray** features for workflow execution and automation
- **OpenShift** add-ons for deployment automation, resource management and GPU and networking support
- Enhancements to underlying **communications libraries**

Project	Features	Value add
CodeFlare	Training toolkit	Packaging and automation of training stack with PyTorch and Ray
	Transfer learning and fine-tuning toolkit	Packaging and automation of downstream tasks with Ray
	Pipelines	Automation API for pipeline scaling and execution
	Data loader	Python native key value store for high throughput GPU data loading
PyTorch	TorchX extension and OpenShift integration	Simplified and unified out of the box deployment of distributed training pipelines
	Transformer architecture extensions	Extending transformer architecture support for new use cases and data modalities
RAY	Ray Core (including Workflows)	Extending execution and scaling of workflow steps using with Ray
	Serving	Contributing to integration of Ray Workflows and Serve to support end-to-end model cycle
	DAG generation	Generation and automation of execution graphs for workflows
	KubeRay	Enhancing OpenShift support for toolkit to run Ray applications on Kubernetes
OPENSHIFT	Scalability and performance enhancements	Multi-NIC Container Networking (CNI), GPU tune profile, topology operator
	Advanced resource and job management add-ons	scheduling and job management capabilities on OpenShift to support concurrent jobs
	Deployment automation extensions	Extending deployment automation for networking configuration and GPU support
NVIDIA NCCL	Tools to capture, analyze, and improve performance of training and inference	Extensions to NVIDIA tools that provide deeper understanding of workloads and resource utilization

Foundation model cluster

A100 GPUs
4x100 Gbps ethernet