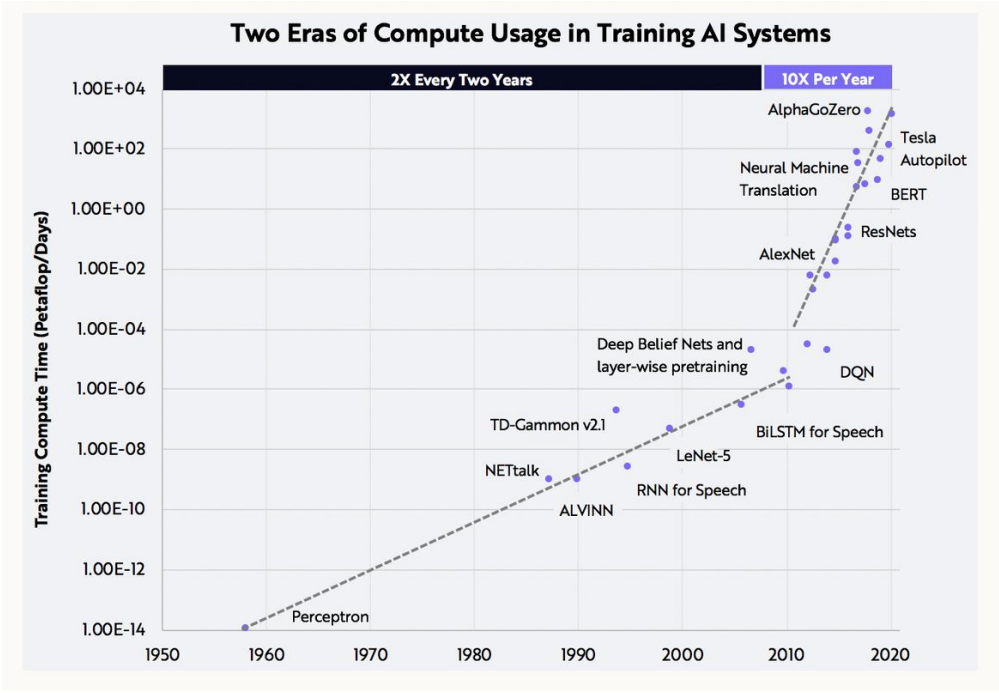
AI-SIG Introduction

Improve AI workloads efficiency and resource utilization

June 30, 2021

Background

- AI market cap $1 trillion today, forecast to reach $30 trillion by 2037, with 21% CAGR, exceeds Internet.*
- Deep Learning performance continuously improves as training dataset increases, but deep learning training (DLT) is compute-intensive
- Model size grows rapidly, training time increases 10X per year
- AI workloads move to cloud and edge computing (training & inferencing)
- Cloud infrastructure needs to be tailored for AI workloads



*A " Petaflop-Day" refers to performing a quadrillion operations per second for a day.

Source: ARK Investment Management LLC, "AI and Compute." OpenAI, https://arkinv.st/2ZOH2Rr.

*Source: ARK investment Management LLC

AI Stack in Public Cloud

AI Service
vision, speech, translation, text analytics, search ...

AI Framework
Tensorflow, Pytorch, Mxnet …

AI Infrastructure
Data: Data Lake, SQL/NoSQL
Compute: Data processing, batch processing, VM/containers/serverless

AI Tools
(Notebooks, IDE, MLOps)

- AI Platform Features
  - Data Integration, e.g. import, preprocessing, store
  - Framework support, e.g. Tensorflow, PyTorch, MXnet
  - Hyperparameter auto tuning
  - Distributed training
  - Online serving
  - Pretrained model, or basic services (Translation/image recognition)
  - Model/results life cycle management (MLops)
  - Infrastructure, resource management
- Typical Platform
  - AWS SageMaker, IBM Watson Studio (IDE and Services)
  - Tensorflow, H2O (Framework)
  - Kubeflow, Volcano (Kubernetes Eco-system)
  - Spark (Data processing)

Challenges using DL at Scale*

- Distributed training
  - Provisioning jobs (gang scheduling, container setup, MPI)
  - Fault tolerance and auto-scaling
- Hyperparameter search
  - Explore vs exploit (early stopping, resource budget per trial)
  - Dynamic resource allocation + distributed training
- End-to-End Workflow
  - Moving data between stages (locality, materialization, random access)
  - Minimizing training/serving skew (unifying the API)
- Stitch framework together
  - Spark (ETL) – Autotune (Hyperparameter search) – Horovod (Distributed training)

*challenges are summarized by Uber Deep Learning Team, reference talk https://www.youtube.com/watch?v=rEB3NPUoxMM

Project Alnair

- Goal: Improve AI workloads efficiency and user experience
- Problems to solve
  - Resource (GPU) monitoring, usage prediction, sharing
  - Distributed training framework integration and improvement
  - Intelligent allocation/scheduling algorithms, considering workloads requirements (e.g., batch scheduling, re-scheduling, preemption, multiple pod templates ..)
  - ML framework optimization to improve time-to-accuracy
- Reference project
  - Kubernetes, Horovod, Ray

Kubernetes

- De-facto Cloud OS
- Powerful container orchestration platform, highly extensible
- Flexible device plugin design to manage heterogeneous resources (GPU, FPGA, NIC)
- Open source: CNCF

- Adoption Increases during pandemic*
  - Role expanding: AI workloads, infrastructure management

*https://www.purestorage.com/content/dam/pdf/en/analyst-reports/ar-portworx-pure-storage-2021-kubernetes-adoption-survey.pdf?utm_source=thenewstack&utm_medium=website&utm_campaign=platform

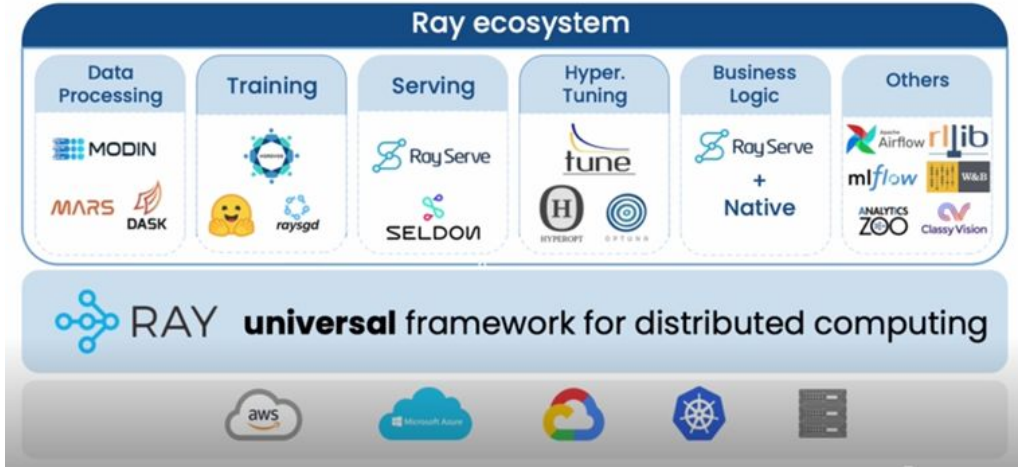
Horovod

- Data-parallel framework for distributed deep learning
- Features
  - Framework agnostic: TensorFlow, Keras, PyTorch, MXNet
  - High performance features: NCCL, GPUDirect, RDMA, tensor fusion
  - Easy to install and use
  - Open source: Linux Foundation AI foundation
- **Ludwig**: DL toolbox allows users to train and test DL models without writing code

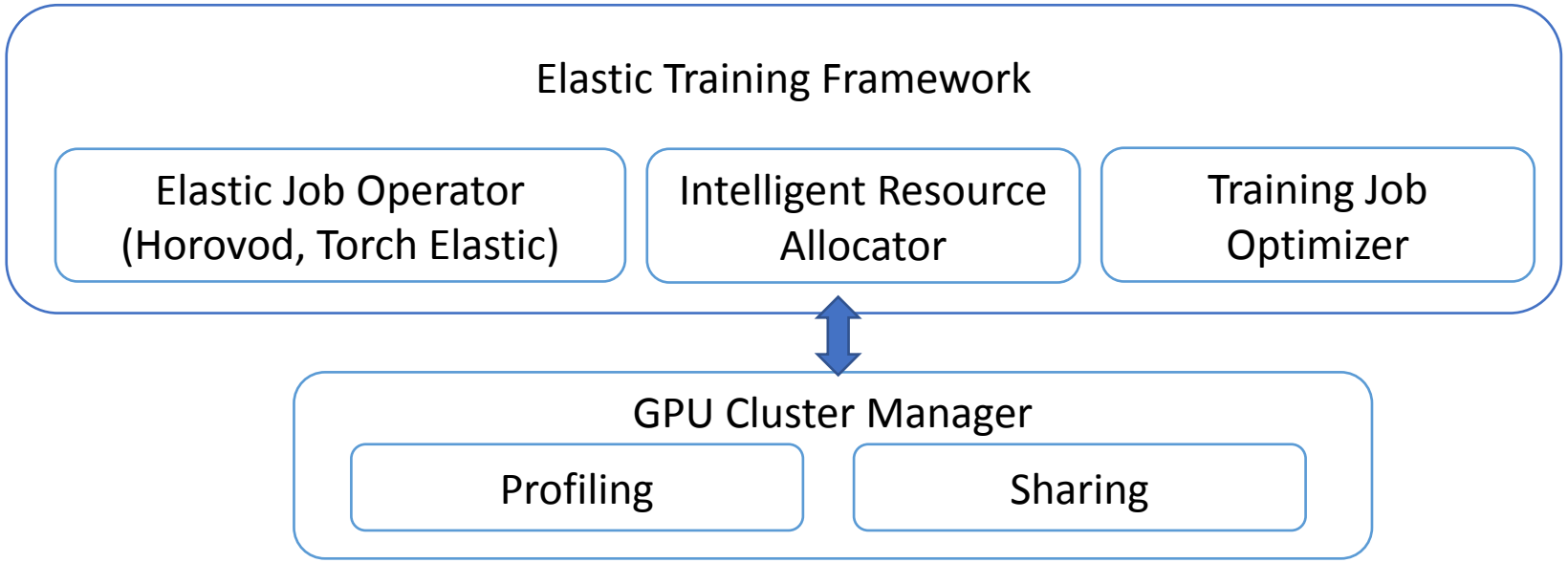
Ray

- Universal Framework for distributed computing
- Features
  - Run anywhere (Kubernetes, cloud, local)
  - General set of distributed compute primitives
  - out-of-the-box ML infra (Ray Tune, RL lib)

- Enables unified infrastructure across all stage of DL workflow



Alnair Architecture



Feature list

- Infrastructure
  - Resource (GPU) utilization monitoring, pattern recognition, usage forecasting
  - Elastic resource allocation
  - Continuous resource utilization optimization with intelligent allocation algorithm
  - Fault tolerance
- ML Framework
  - Hyper parameter tuning (parameter selection, early termination)
  - Data/model parallelism optimization (GPU topology, network latency)

Current Progress

- Implement two components running in Kubernetes/Arktos Cluster
  1. GPU utilization Profiler (Daemon Set)
     - Nvidia dcgm-exporter to collect GPU metrics (GPU/memory/power/temperature …)
     - Prometheus to store and query time series metrics data
     - Metrics analyzer to identify training job resource utilization pattern, tag results as annotations to nodes in the cluster
  2. Elastic-training framework (CRD and Operator)
     - ElasticHorovodJob CRD to define the user image and resource requirements
     - ElasticHorovodJob Controller to scale job up and down with no interruption, provide fault tolerance feature
     - GPU allocator to dynamically allocate GPUs for ElasticHorovodJob; the number of GPUs is assigned within min-max range based on current availability

Questions ?