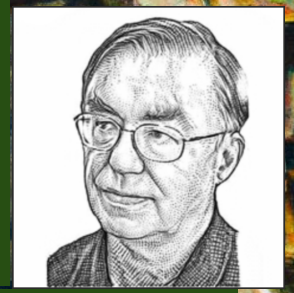
Integrated Systems for Deep Learning and Data Engineering on Clouds and HPC Systems



10th Multicore World ,13 to 17 February 2023

Wellington, New Zealand

Geoffrey Fox, Biocomplexity Institute and Computer Science Department

University of Virginia UVA

UVA: Arup Sarker, Kaiying Shan(UVA UG graduated), Mills Staylor, Tianle Zhong

Indiana University/Sri Lanka (Moratuwa): Niranda Perera, Damitha Lenadora, Supun Kamburugamuve, Thejaka Amila Kanewela, Chathura Widanage

Rutgers/ Brookhaven: Shantenu Jha

DALL-E

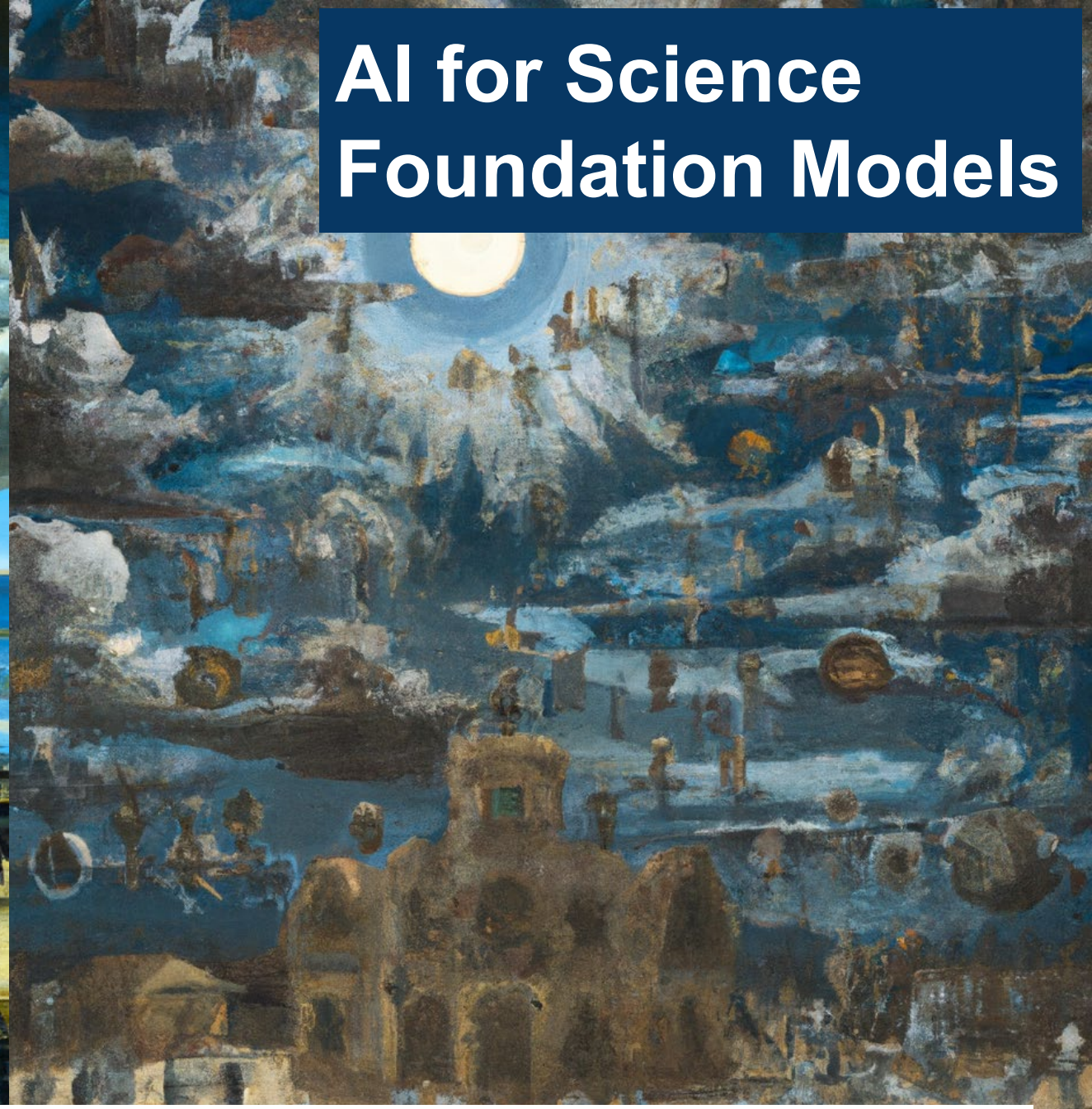
A beautiful painting of a library with lots of books surrounding a giant computer chip in style of M.C. Escher, John Constable

Abstract

- AI, as seen today by large-scale deep learning, presents complex challenges to computer systems. It requires the adaptive execution of heterogeneous components, each of which is a cluster of parallel tasks.
- Further, large amounts of data need to be read, written, and often moved in small dynamic batches. Deep learning must be efficiently linked to pre and post-processing (data engineering). This implies converged environments with Java, Python, and C++ ecosystems.
- Further, AI must be integrated with better-understood large parallel simulations. The simulations can invoke AI to control passage through phase space (often with ensembles). Alternatively, it can train surrogates used to replace all or part of the simulation. In the latter case, there are consequent inferences, as in computational fluid dynamics or climate simulations where AI can learn microstructure.
- This implies we must support systems that run well in conjunction with classic Slurm-MPI-based simulations as well as in modern cloud environments, including challenges from shared resources due to multi-tenancy. This extends the needed convergence to link HPC and cloud environments.

Deep Learning and Data Engineering

AI for Science Foundation Models



DALL-E

A beautiful painting of ten small black robots swimming in the sea with a supercomputer on an island by Constable

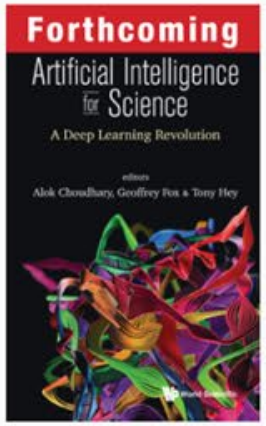
A beautiful painting of sixteen small robots outside a gorgeous palace in stormy cloudy sky in style of Leonardo da Vinci. There is a Sun and a Moon in the sky

World Scientific
Connecting Great Minds

https://doi.org/10.1142/13123

Search | My Cart | Sign in

Subject | Journals | Books | Major Reference Works | Resources For Partners | Open Access | About Us | Help



Artificial Intelligence for Science

A Deep Learning Revolution

Save for later

https://doi.org/10.1142/13123 | January 2023
Pages: 500

Edited By: Alok Choudhary (*Northwestern University, USA*), Geoffrey Fox (*University of Virginia, USA*) and Tony Hey (*Rutherford Appleton Laboratory, UK*)

ISBN: 978-981-126-566-2 (hardcover) | GBP 140.00

Tools | Share | Recommend to Library

Description | Authors

This unique collection introduces AI, Machine Learning (ML), and deep neural network technologies leading to scientific discovery from the datasets generated both by supercomputer simulation and by modern experimental facilities.

Huge quantities of experimental data come from many sources — telescopes, satellites, gene sequencers, accelerators, and electron microscopes, including international facilities such as the Large Hadron Collider (LHC) at CERN in Geneva and the ITER Tokamak in France. These sources generate many petabytes moving to exabytes of data per year. Extracting scientific insights from these data is a major challenge for scientists, for whom the latest AI developments will be essential.

The timely handbook benefits professionals, researchers, academics, and students in all fields of science and engineering as well as AI, ML, and neural networks. Further, the vision evident in this book inspires all those who influence or are influenced by scientific progress.

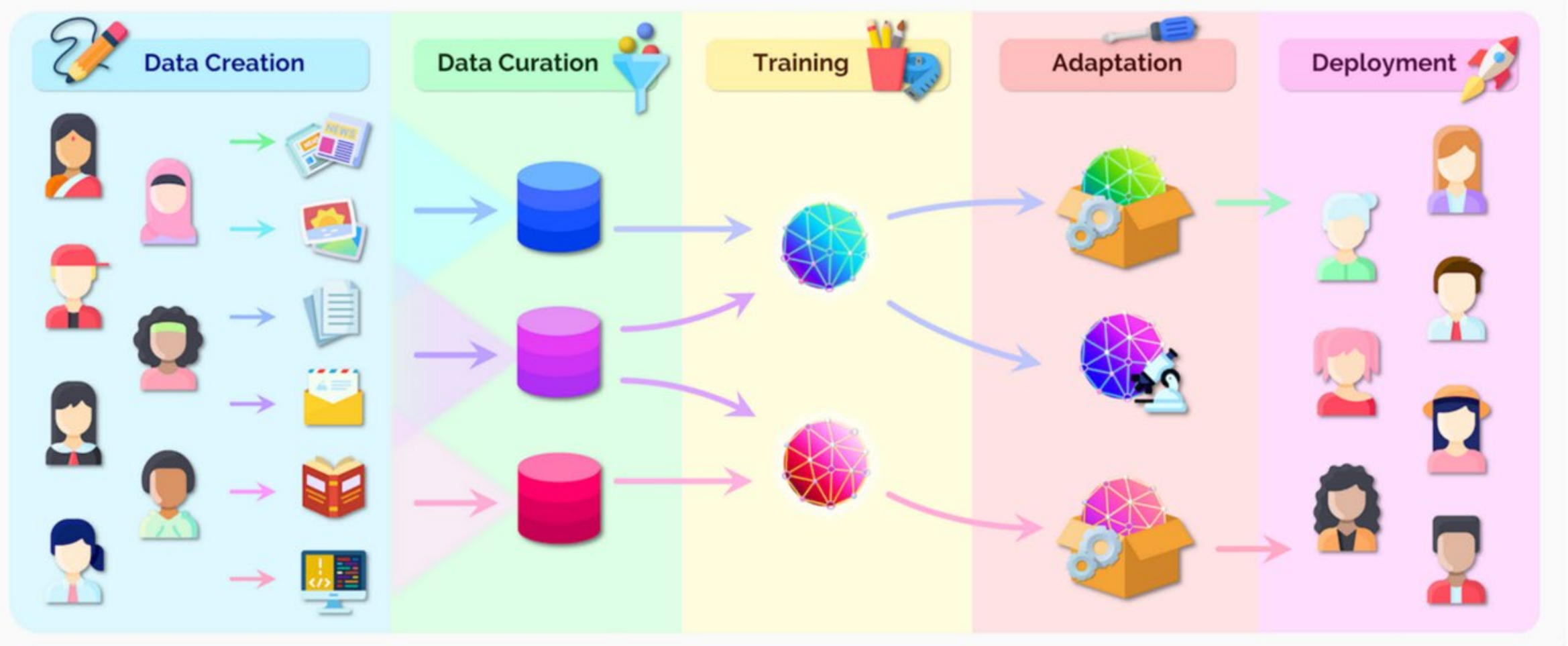
40 Chapters
95% of AI Discussed is Deep Learning

Research Plan: AI for Science Patterns

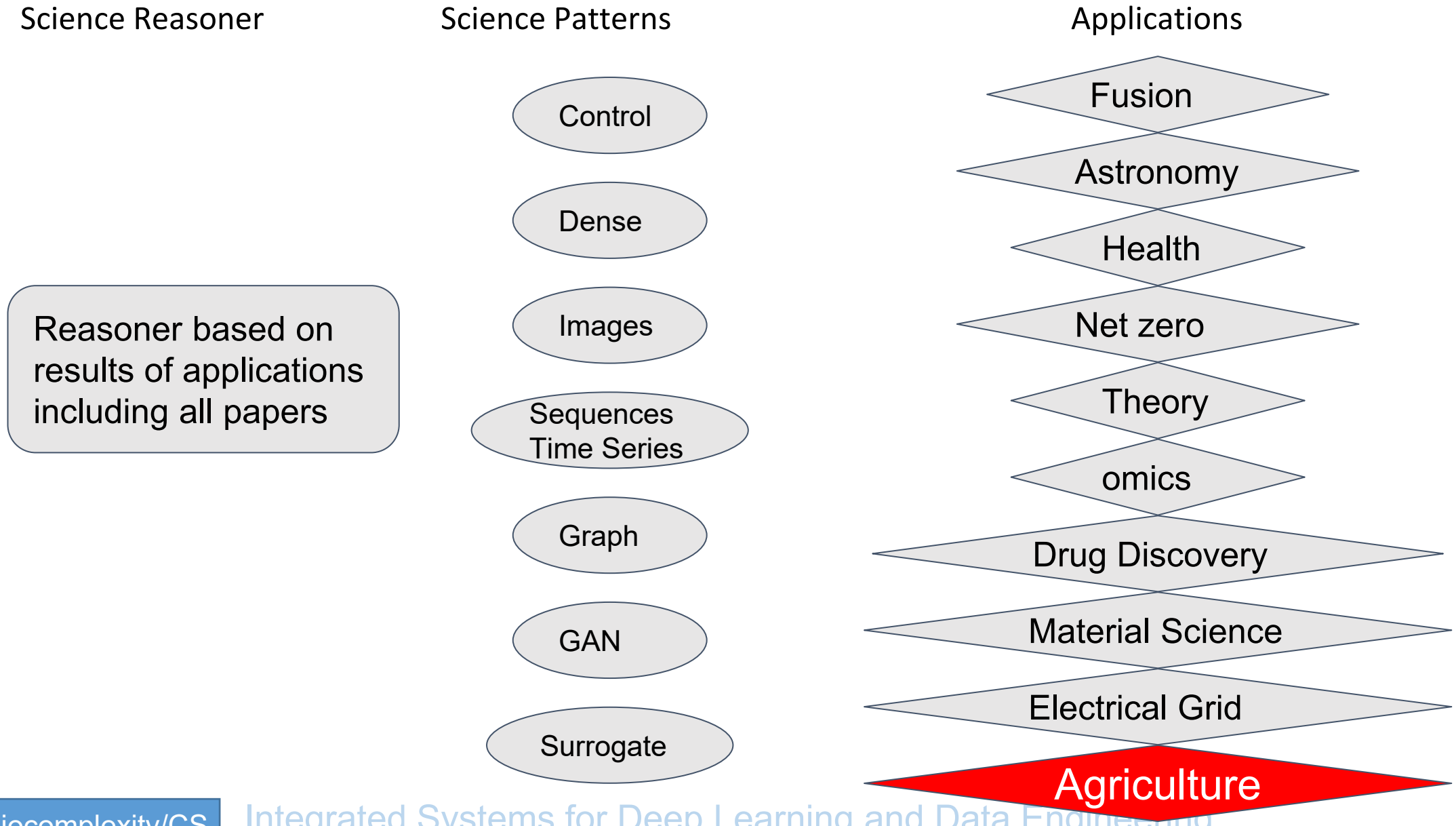
- Currently, **deep learning** plays a dominant role in science, from data analytics and simulation surrogates to policy decisions.
- We can group the system structure into a “handful” (two hands ≤10) of **patterns** such as
- an **Overall Reasoner** based on **reinforcement learning** and large language models to learn the world’s knowledge and control experiments;
- **Image-based systems** for astronomy, pathology, microscopy, and light scattering;
- **Graph-based systems** such as in social media and traffic studies; represent molecular and other structure;
- **Dense systems t**o map structure to properties as in drug discovery;
- **Time series and sequence (Recurrent, Transformer)** models as in language, earth, and environmental science;
- **GAN/Diffusion** models to generate scenarios as in datasets to test experimental system.
- **Surrogate** models have distinctive issues where there is no current consensus
- **All Network types** can be mixed together as in text to image systems DALL-E and Imagen.
- We suggest taking good **examples of each pattern** and **supporting them** with high-performance, easy-to-use environments in end-to-end systems, including data engineering.
- This would cover parallelism, storage and data movement, security, and the user interface. Then we explore multiple examples of each pattern, possibly leading to “**Foundation**” models for each of them.

Big and Foundation Models

- Transfer Learning builds AI model in one area and then uses it with modest additional training in another area
- Foundation Models follow this to an extreme and train on so much data that model can generalize to cover “everything”



Overall AI for Science Architecture



Deep Learning and Data Engineering

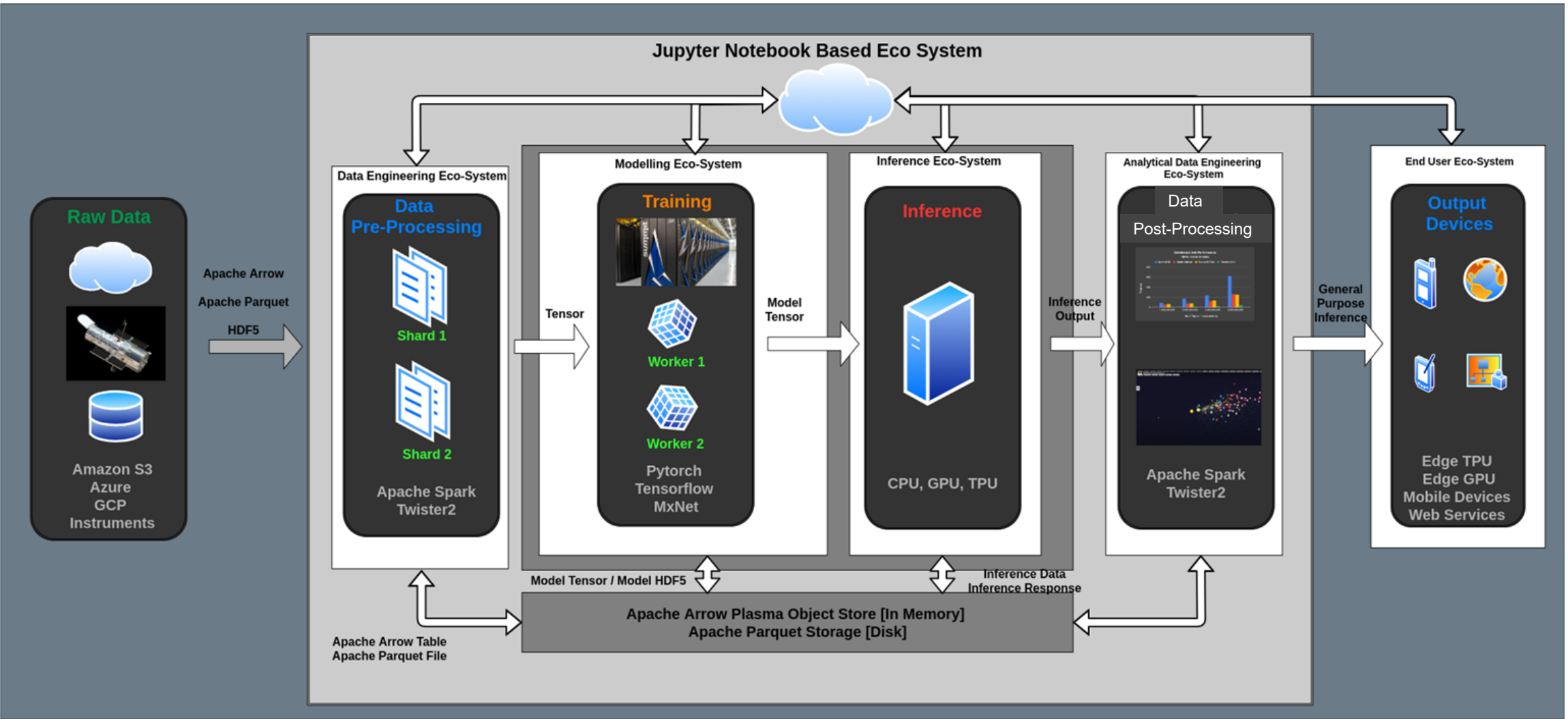
First High Performance Data Engineering
Later High Performance Data Science

A beautiful painting of a library with lots of books surrounding a giant computer chip in style of John Constable

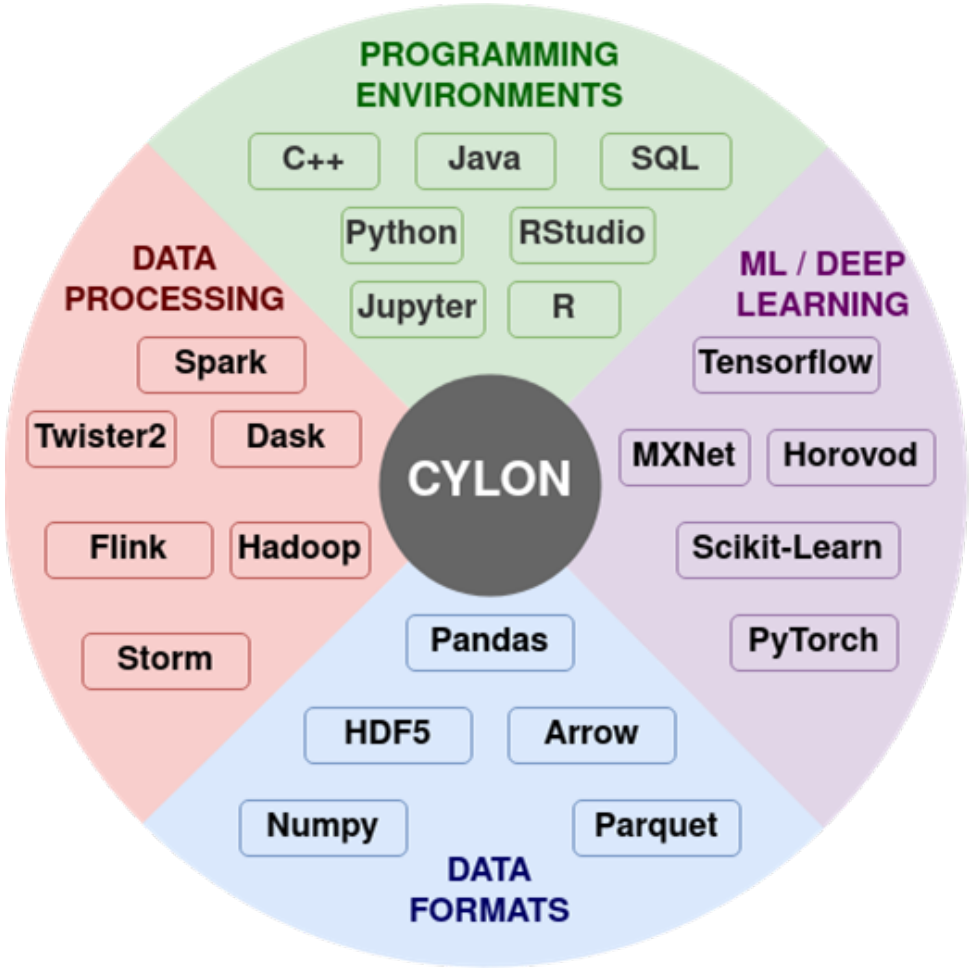
Typical Deep Learning System Environment

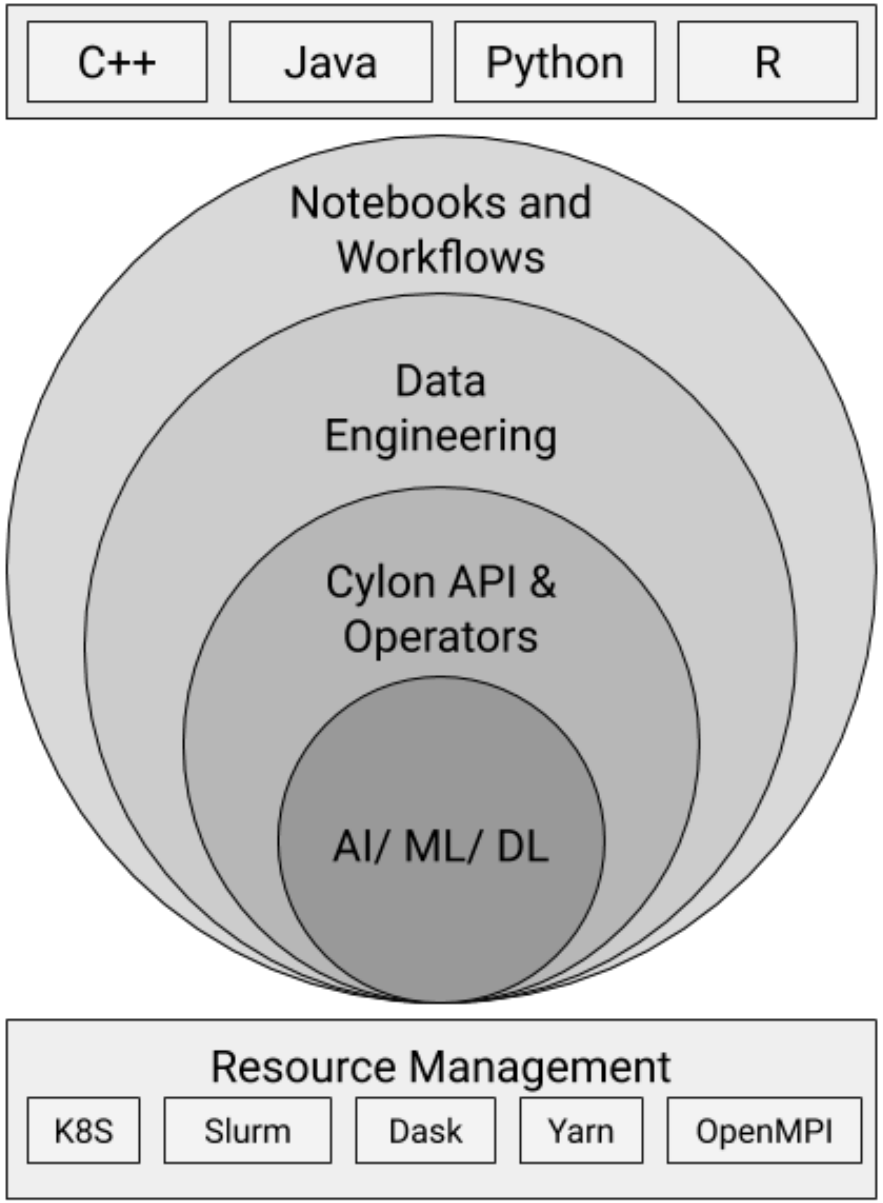
Data Engineering: Data => Information preprocessing -- Hadoop, Spark, Scikit-Learn

Data Science: Information => Knowledge Compute and I/O intensive step (PyTorch and Tensorflow)



CYLON



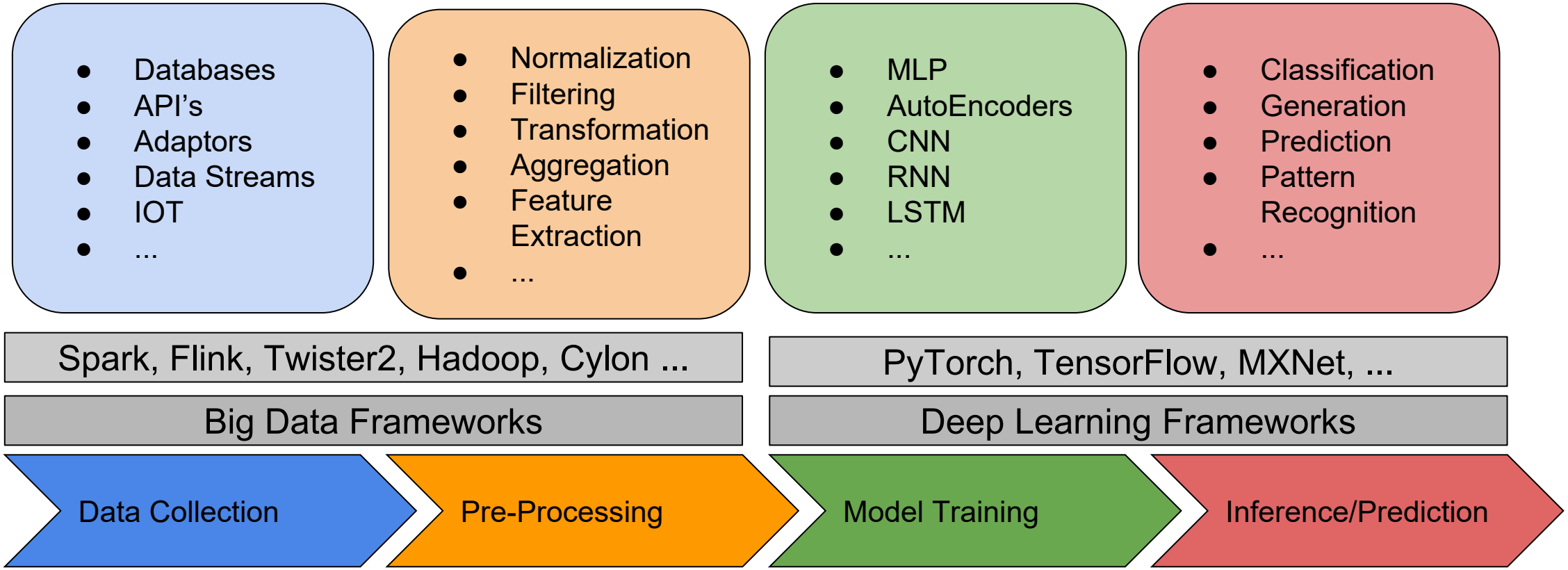


Originally C++ Kernel for Java Systems like Spark, Flink, Twister2
Now standalone from Python

Data Engineering (DE) and Deep Learning (DL)

Java systems have difficulties in linking to Python/C++ with high performance

From Data Management(DM) to Data Engineering (DE) to Deep Learning (DL)



There is Data Engineering in **Java** and **Python/R** ecosystems

Must support Parallelism as Automatically as Possible

- At a low level, parallelism will be
  - NCCL MPI Horovod etc.
  - or pleasing parallelism (many task)
- But the user can be shielded from hard truly parallel cases by **libraries**
- Originally these libraries were Linear Algebra for **simulations** and implemented in programming models such as High Performance Fortran HPF, C++, Java
- **Data Analytics** has developed a very powerful set of such operator/function libraries with
  - Pandas and Numpy **array, table and dataframe** operations
  - Deep Learning in PyTorch and Tensorflow
  - Spark transformations
- Modin has described architecture of Pandas Operators
- Further this **is proxied** with Python frontends and can invoke parallel (C++) implementations unbeknownst to user
- So goal is to develop infrastructure for **operator-based parallelism**
- Uses approach similar to "**High Performance Fortran**" developed by Ken Kennedy at Rice University 1990-2000

Some Possibly Parallel Operators/Functions

- **Classic Parallel Computing: (720 MPI functions)**
  - AllReduce, Broadcast, Gather, Scatter
- **Linear Algebra (320 functions in SCALAPACK at one precision)**
  - Matrix and Vector Operations
- **Tables 224 Pandas operators for Dataframe out of 4782 total**
  - **Intersect:** Applicable on two tables having similar schemas to keep only the records that are present in both tables.
  - **Join:** Combines two tables based on the values of columns.Includes variations Left, Right, Full, Outer, and Inner joins.
  - **OrderBy:** Sorts the records of the table based on a specified column.
  - **Aggregate:** Performs a calculation on a set of values (records) and outputs a single value (Record). Aggregations include summation and multiplication.
  - **GroupBy:** Groups the data using the given columns; GroupBy is usually followed by aggregate operations. Famous from **MapReduce**
- **Arrays (1085 Numpy)**
- **Tensors (>700 Tensorflow, PyTorch, Keras)**
  - All the myriad of Numpy array operations
  - Add a layer to a deep learning network
  - Forward (calculate loss) and backward (calculate derivative) propagation
  - Checkpoint weights of network

Cylon: A High Performance Distributed Data Table

- **Cylon** is a high performance C++ kernel and a distributed runtime for **big data processing supporting operator parallelism**
  - Built around **Apache Parquet and Apache Arrow** based storage and in-memory data structures which offer high performance and zero-copy where possible
    - Supports integration with Deep Learning workloads, Pandas and Numpy
    - Zero-Copy data transfer between heterogeneous systems and languages.
- **Table API**, an abstraction for ETL (extract, transform, load) for scientific computing and deep learning workloads including Pandas, HDF5
  - Join, Union, Intersect, Difference, Product, Project … ~40 operators
- **Written in C++**, APIs available in Java and Python (via Cython).
- Due to Arrow, Cylon faster sequentially than Pandas in many cases
- Also links to **Parsl**, **Radical Pilot**, **Dask** and **Ray** efficiently so can support **asynchronous mixtures** (**islands**) of MPI (**Bulk Synchronous Processing**) tasks
  - Does not require classic MPI engine so runs inside or outside Slurm
- Future: using **MLIR**, support **graph of asynchronously linked parallel operators**

Distributed Dataframe Operator Patterns

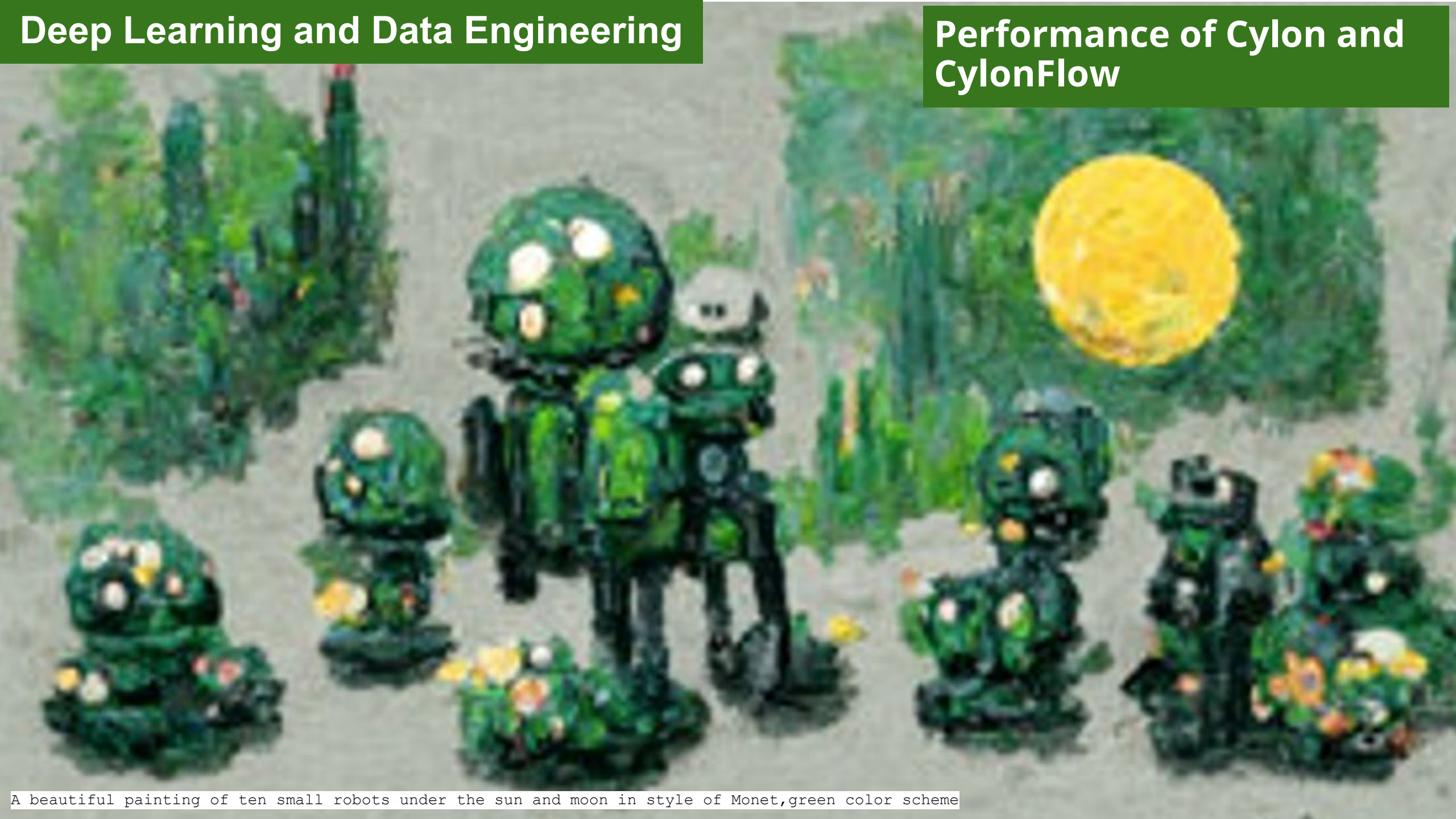
- All Pandas Operators supported by 8 patterns of parallel algorithms so can extend to all 224

Pattern		Operators	Result Semantic	Communication
Embarrassingly parallel		Select, Project, Map, Row-Aggregation*	Partitioned	-
Loosely Synchronous	Shuffle Compute	Union, Difference, Join, Transpose	Partitioned	Shuffle
	Sample Shuffle Compute	Sort	Partitioned	Gather, Bcast, Shuffle, AllReduce
	Combine Shuffle Reduce	Unique, GroupBy	Partitioned	Shuffle
	Broadcast Compute	Broadcast-Join**	Partitioned	Bcast
	Globally Reduce	Column-Aggregation*	Replicated	AllReduce
	Halo Exchange	Window	Partitioned	Send-recv
I/O (Partitioned read/write)		Read/Write*	Partitioned	Send-recv, Scatter

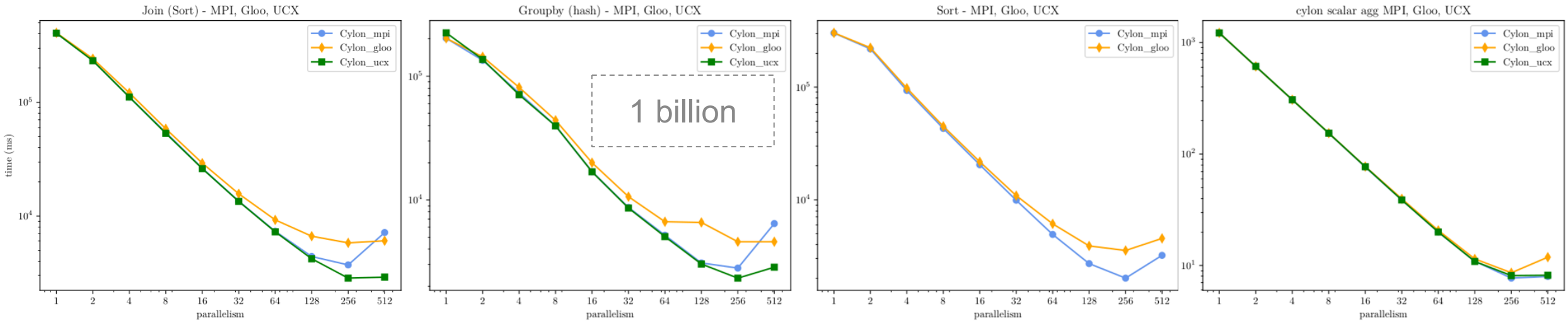
*Not categorized in Modin, **Specialized join algorithm

Deep Learning and Data Engineering

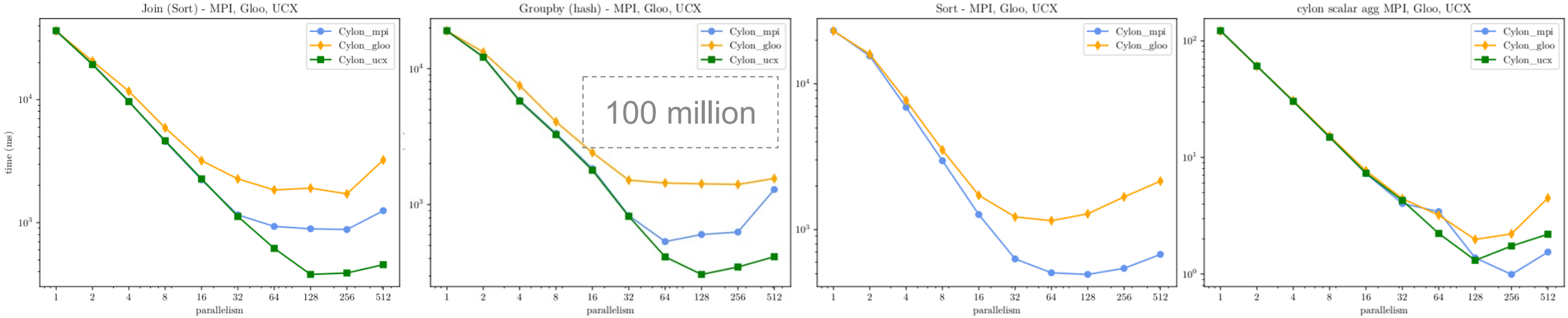
Performance of Cylon and CylonFlow



Communication Library Performance - Strong Scaling



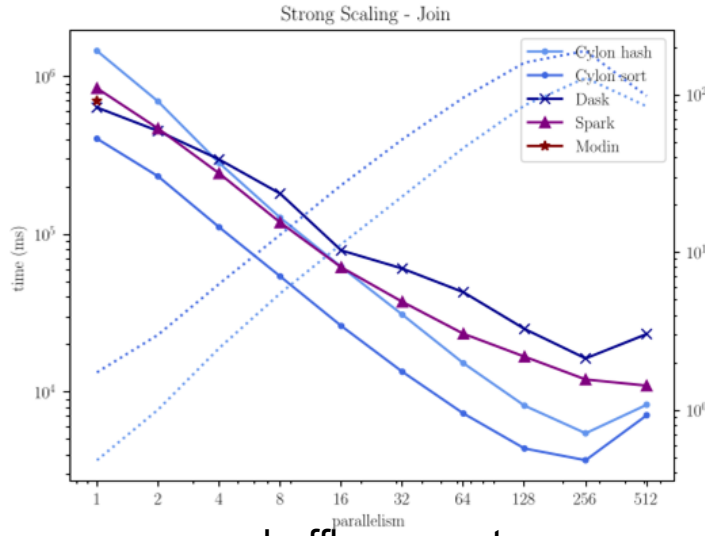
shuffle compute | combine shuffle compute | sample shuffle compute | globally reduce



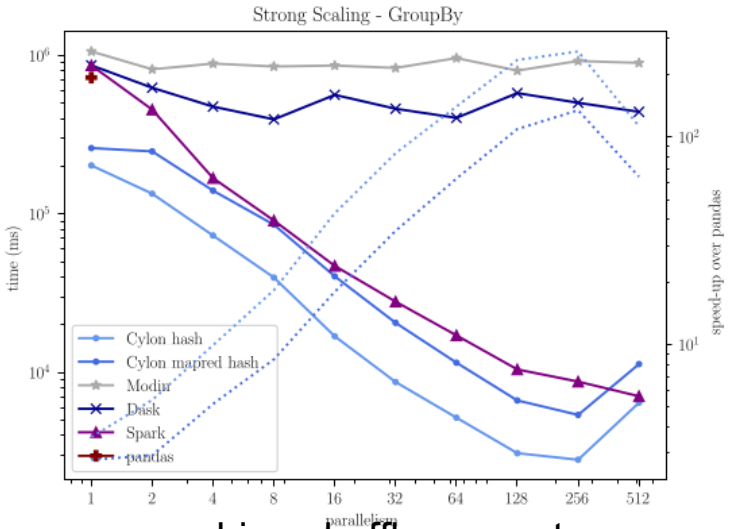
Top row a billion data points, bottom row 100 million

Parallel Operator Performance - Strong Scaling

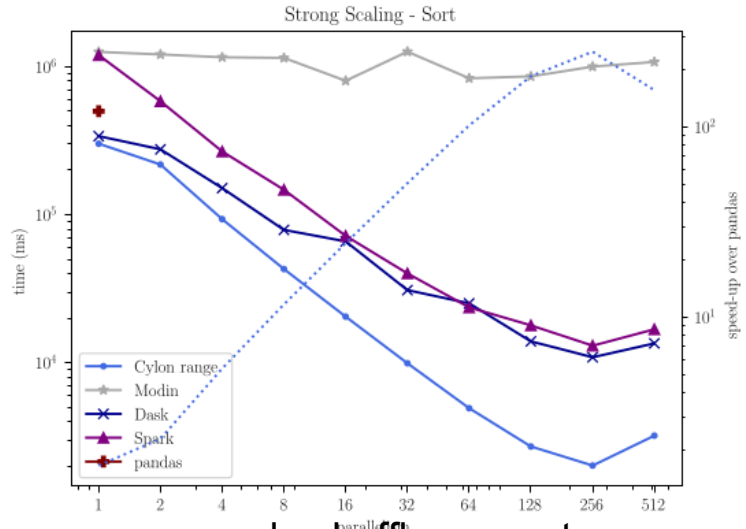
Dashed blue lines are speedup for Cylon implementations



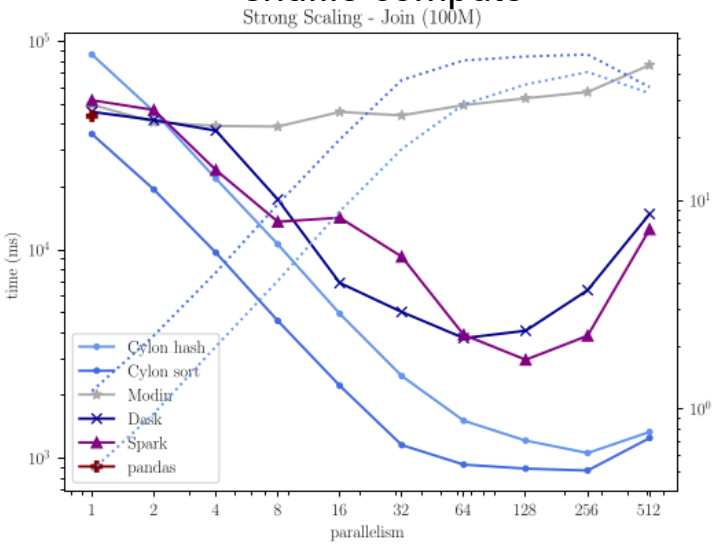
shuffle compute

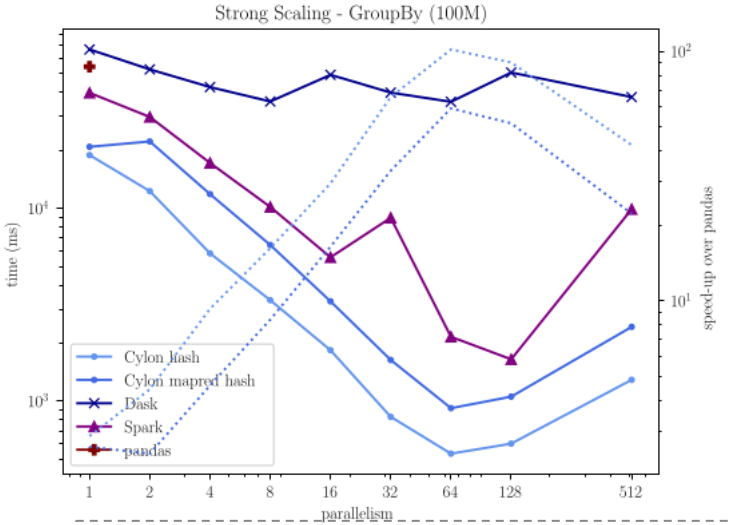


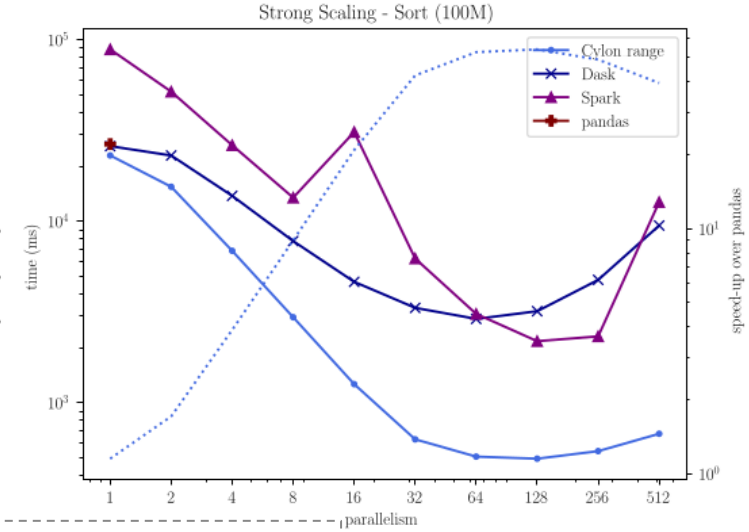
combine shuffle compute



sample shuffle compute

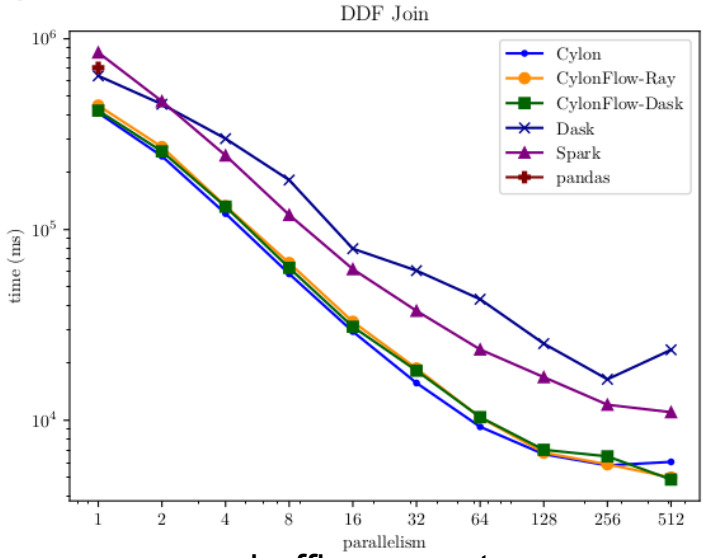




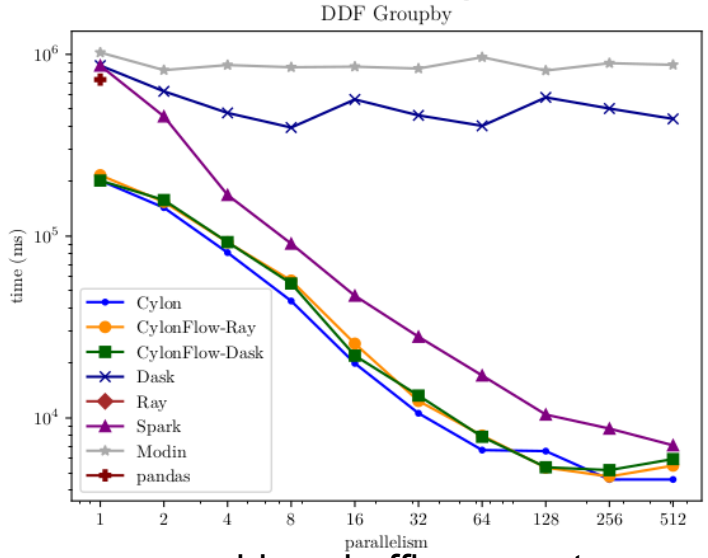


Top row a billion data points, bottom row 100 million

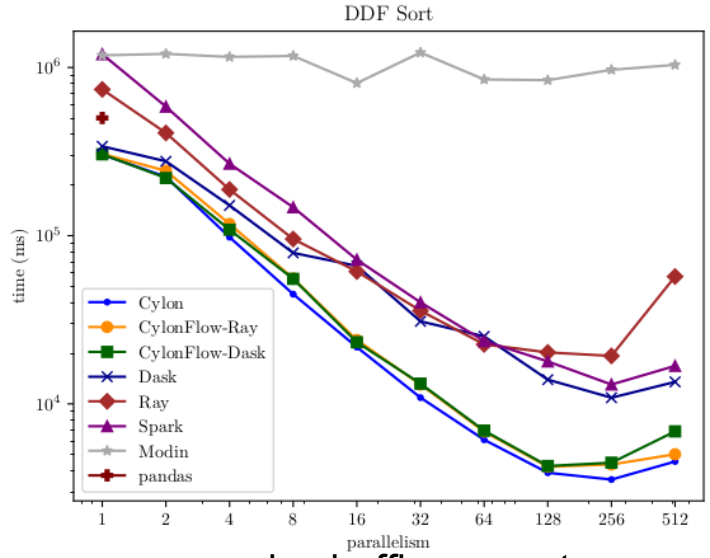
CylonFlow on Dask & Ray



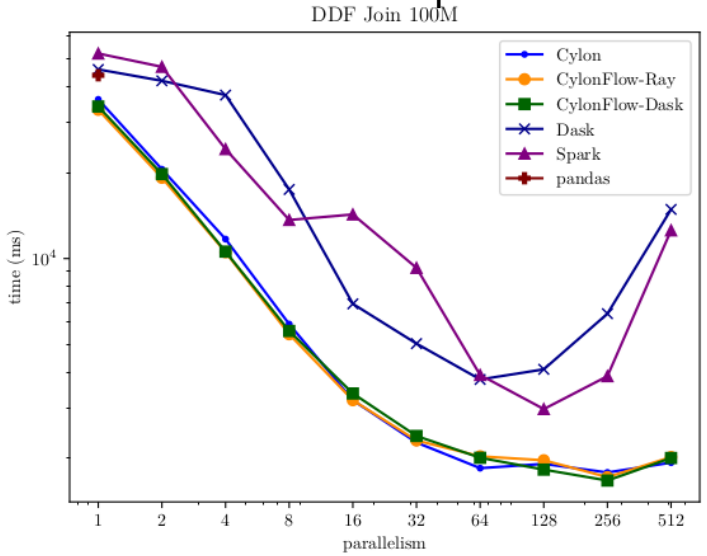
shuffle compute

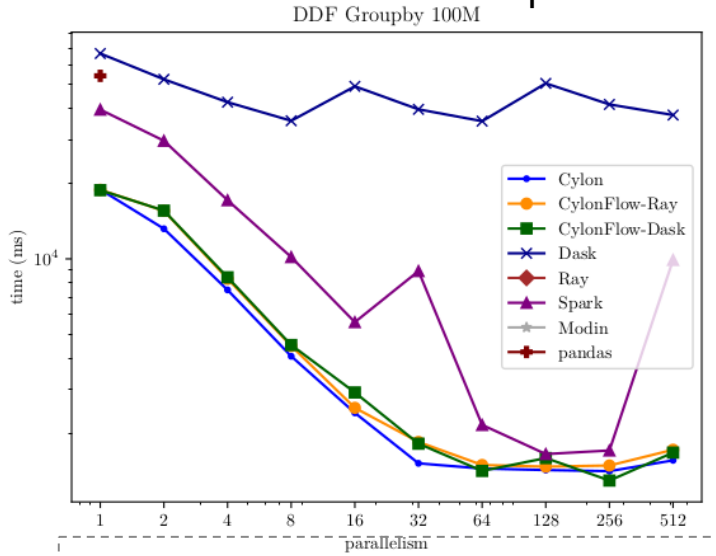


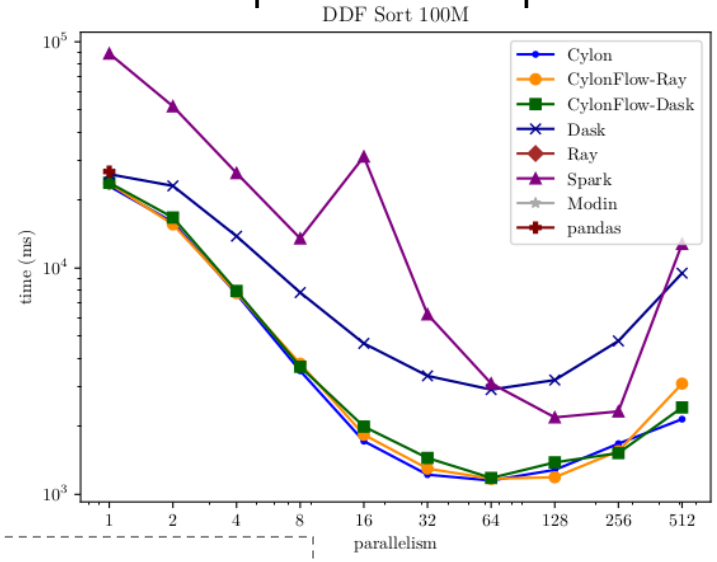
combine shuffle compute



sample shuffle compute

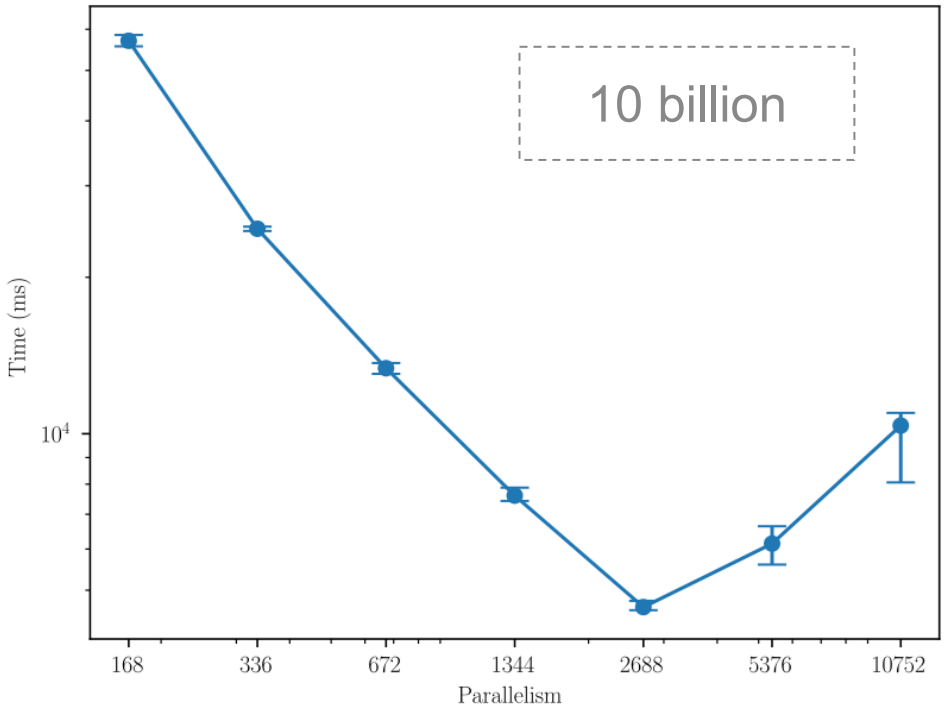


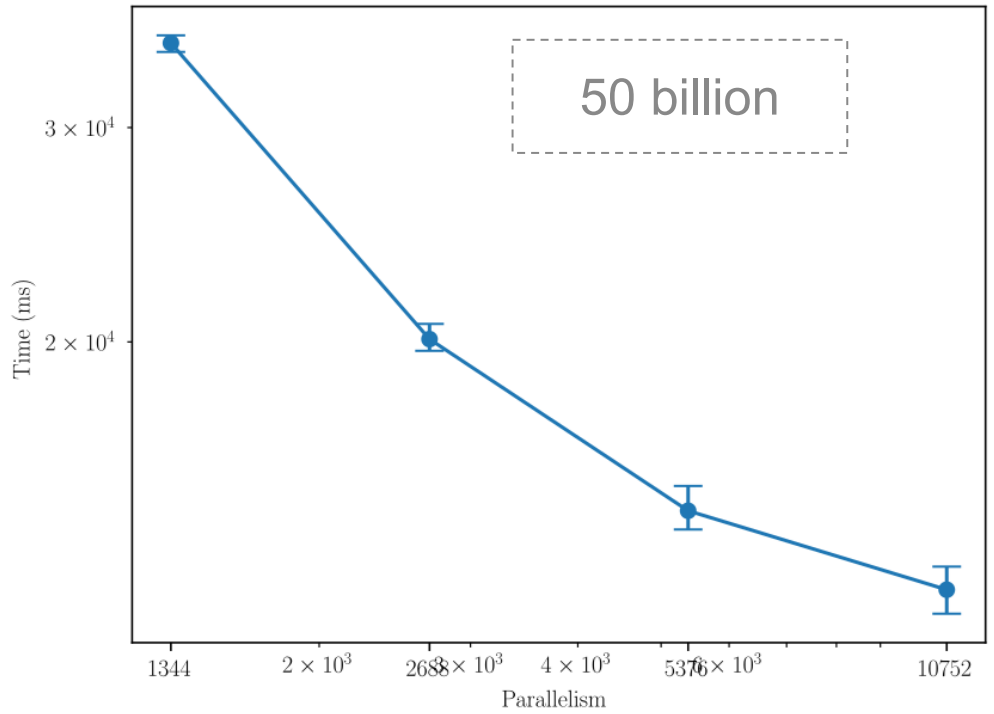




Top row a billion data points, bottom row 100 million

Summit Results - Strong Scaling

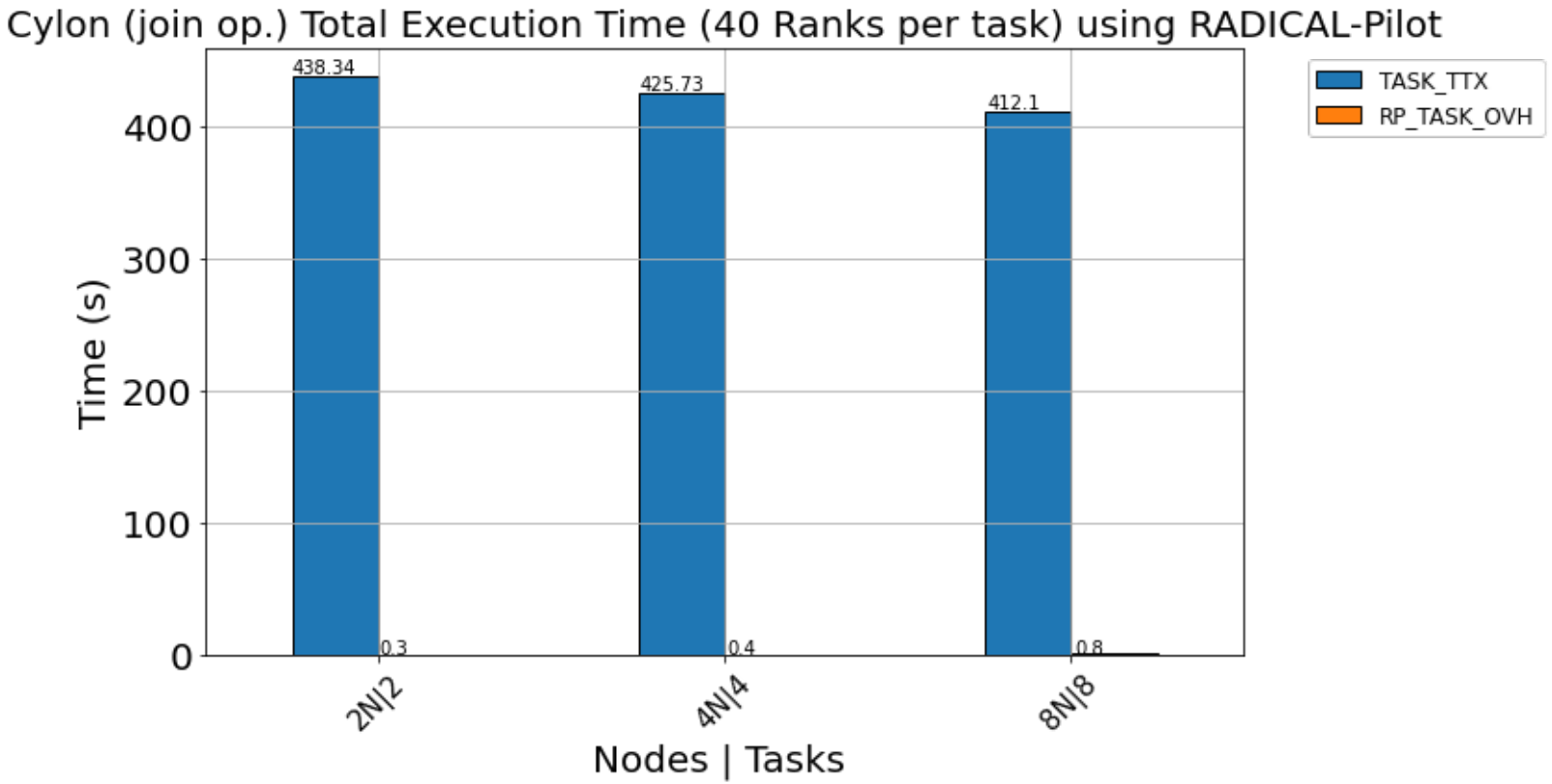




- Communication cost decreases as problem size increases or computer size decreases
- For 50 billion data points → inflection point occur at parallelism > 10k

Cylon and Radical Pilot on Summit

Weak Scaling upto 320 way parallelism



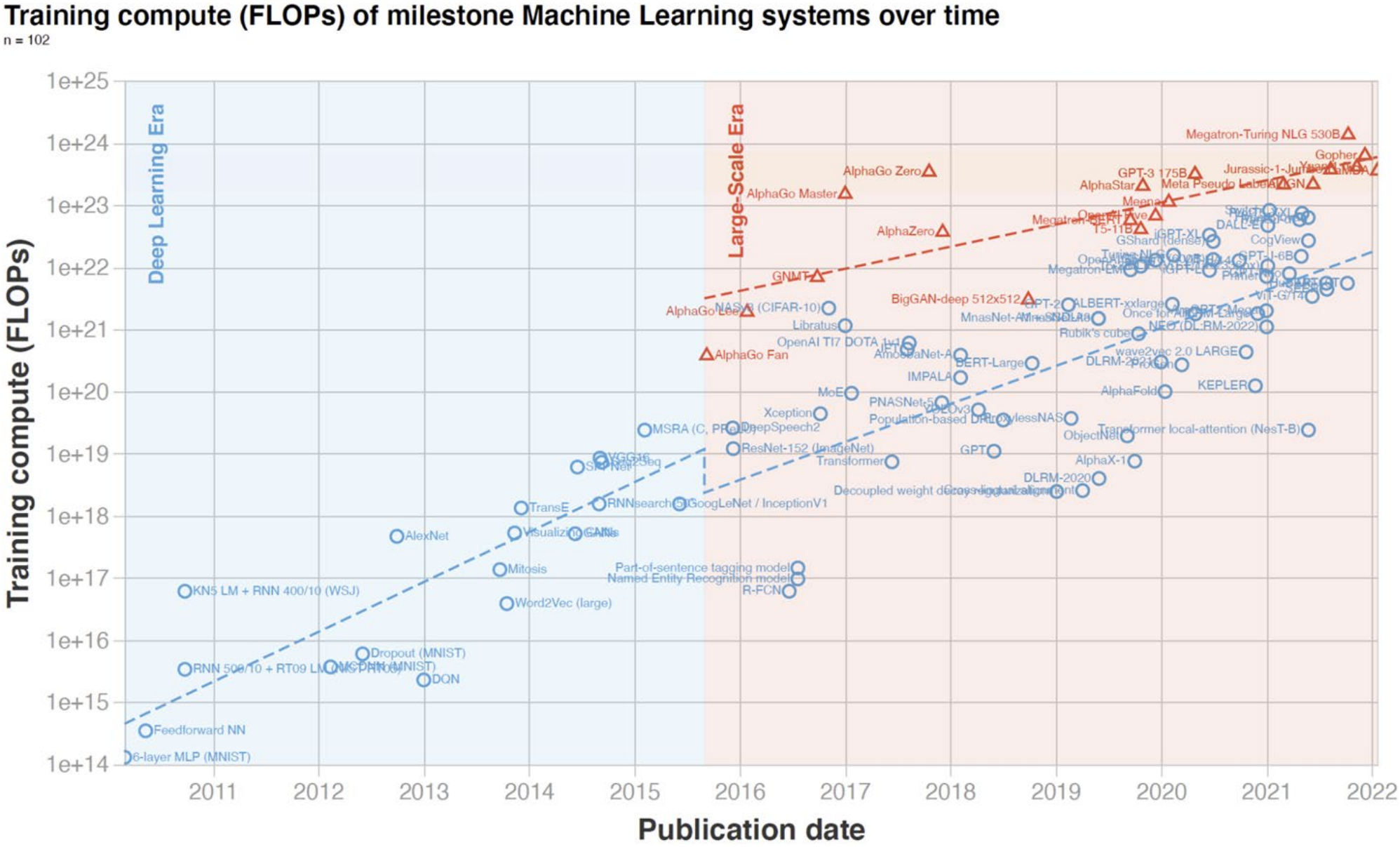
Assume can use HPC workflow or Ray distributed OS with parallel operators for Pandas Numpy and Deep Learning

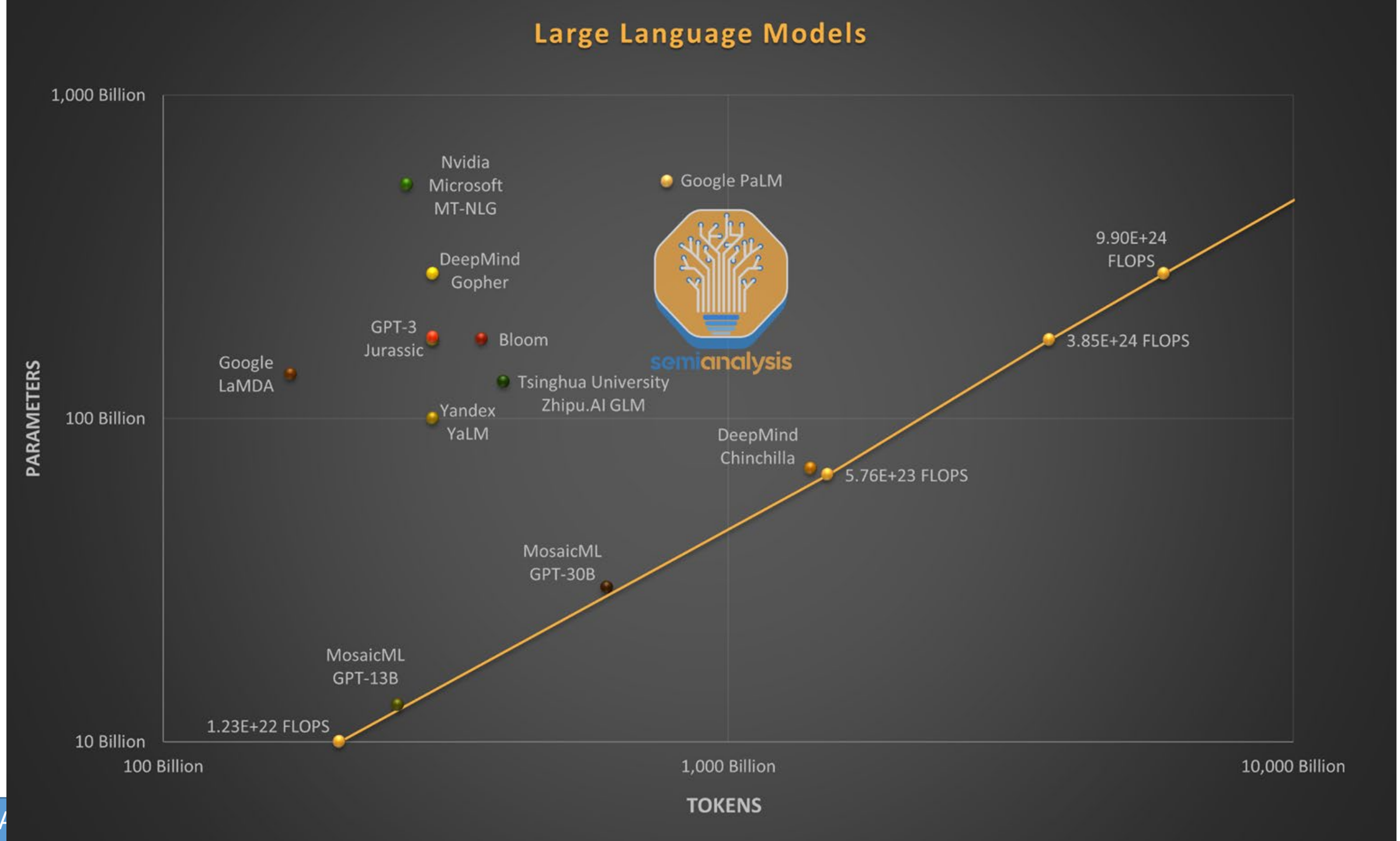
Deep Learning and Data Engineering

A beautiful picture of students texting on iphones seated in a circle . The glowing sun appears as a giant computer chip with clouds in the sky. in style of Monet

Deep Learning Performance

State of the Art AI Models are increasing in Model and Data size




Large Language Models
1,000 Billion
100 Billion
10 Billion
PARAMETERS
100 Billion
1,000 Billion
10,000 Billion
TOKENS
Nvidia
Microsoft
MT-NLG
Google PaLM
DeepMind
Gopher
GPT-3
Jurassic
Bloom
Google
LaMDA
Tsinghua University
Zhipu.AI GLM
Yandex
YaLM
DeepMind
Chinchilla
MosaicML
GPT-30B
MosaicML
GPT-13B
1.23E+22 FLOPS
5.76E+23 FLOPS
3.85E+24 FLOPS
9.90E+24
FLOPS
semianalysis

Optimal LLM Training Cost				
Model	Size (# Parameters)	Tokens	GPU	Optimal Training Compute Cost
MosaicML GPT-30B	30 Billion	610 Billion	A100	$ 325,855
Google LaMDA	137 Billion	168 Billion	A100	$ 368,846
Yandex YaLM	100 Billion	300 Billion	A100	$ 480,769
Tsinghua University Zhipu.AI GLM	130 Billion	400 Billion	A100	$ 833,333
Open AI GPT-3	175 Billion	300 Billion	A100	$ 841,346
AI21 Jurassic	178 Billion	300 Billion	A100	$ 855,769
Bloom	176 Billion	366 Billion	A100	$ 1,033,756
DeepMind Gopher	280 Billion	300 Billion	A100	$ 1,346,154
DeepMind Chinchilla	70 Billion	1,400 Billion	A100	$ 1,745,014
MosaicML GPT-70B	70 Billion	1,400 Billion	A100	$ 1,745,014
Nvidia Microsoft MT-NLG	530 Billion	270 Billion	A100	$ 2,293,269
Google PaLM	540 Billion	780 Billion	A100	$ 6,750,000

Multiply by number of hyperparameters (100)

Commercial Applications Training cost << Inference

Science is opposite in many applications although turbulence inference cost could be > surrogate training as inference for each space-time point

Using ChatGPT like search in Inference ($107B/year is current Google Search Cost)

Real Time Query Processing Additional Costs x 3 for other extra costs				
Metric	A100	H100	TPUv4	TPUv5
Query per day	27,648,000,000	27,648,000,000	27,648,000,000	27,648,000,000
Average Cost Per Query	$ 0.0001669	$ 0.0000592	$ 0.0000994	$ 0.0000373
Daily Incremental Cost	$ 4,613,626	$ 1,636,567	$ 2,748,044	$ 1,030,516
Annual Incremental Cost	$ 1,683,973,626	$ 597,346,885	$ 1,003,035,927	$ 376,138,473

Mixture of Experts in GLaM Generalist Language Model

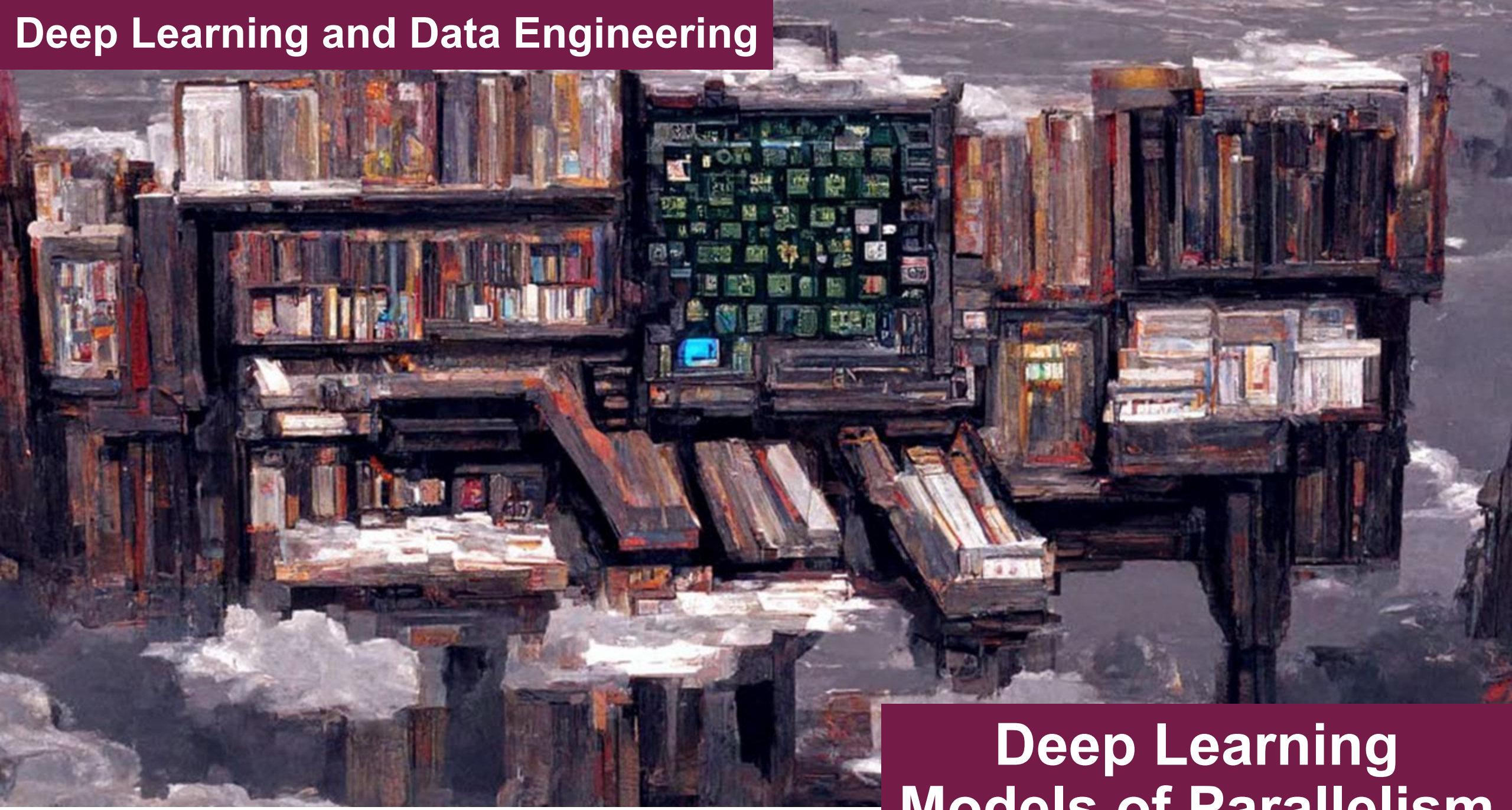
Google's "GLaM: Efficient Scaling of Language Models with Mixture-of-Experts." has 1.2 trillion parameters, which is approximately 7x larger than GPT-3.
It consumed only 1/3 of the energy used to train GPT-3 and required half of the FLOPS for inference while achieving better accuracy.
GLaM only activates 97B parameters (8% of 1.2T) per token during inference. (64 experts)

		GPT-3	GLaM	relative
cost	FLOPs / token (G)	350	**180**	**–48.6%**
	Train energy (MWh)	1287	**456**	**–64.6%**
accuracy on average	Zero-shot	56.9	**62.7**	**+10.2%**
	One-shot	61.6	**65.5**	**+6.3%**
	Few-shot	65.2	**68.1**	**+4.4%**

Performance on MLPerf Training

- Efficiency = Speedup/# Accelerators
- TPUv4 and H100 referred to 4 GPU A100 as unit performance
- Parallel Performance above 1024 GPUs not very good so deep learning systems not huge and occupy small fraction of a supercomputer.
  - Frontier has 37,888 AMD Radeon Instinct MI250X GPUs

ResNet on ImageNet	4 PCIE	4 SXM	8	64	1024	3456	4096
A100	0.92	1	0.95	0.76	0.39		0.16 (4216)
TPUv4						0.28	0.28
H100			1.86	1.52 (32)			
BERT on Wikipedia	4 PCIE	4 SXM	8	64	1024	3456	4096
A100	0.72	1	0.99	0.84	0.31		0.16
TPUv4						0.17	0.18
H100			2.62	2.32 (32)			



A beautiful painting of a library with lots of books surrounding a giant computer chip i

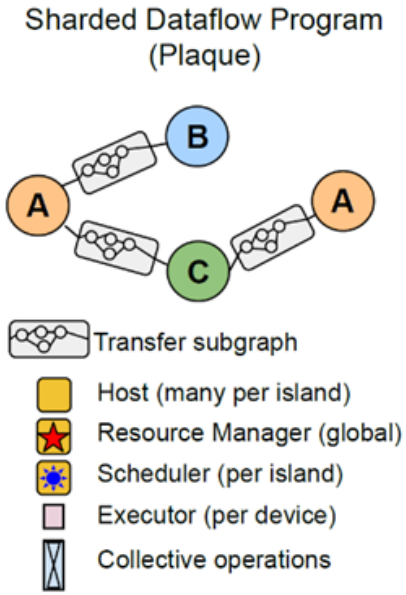
Deep Learning Models of Parallelism

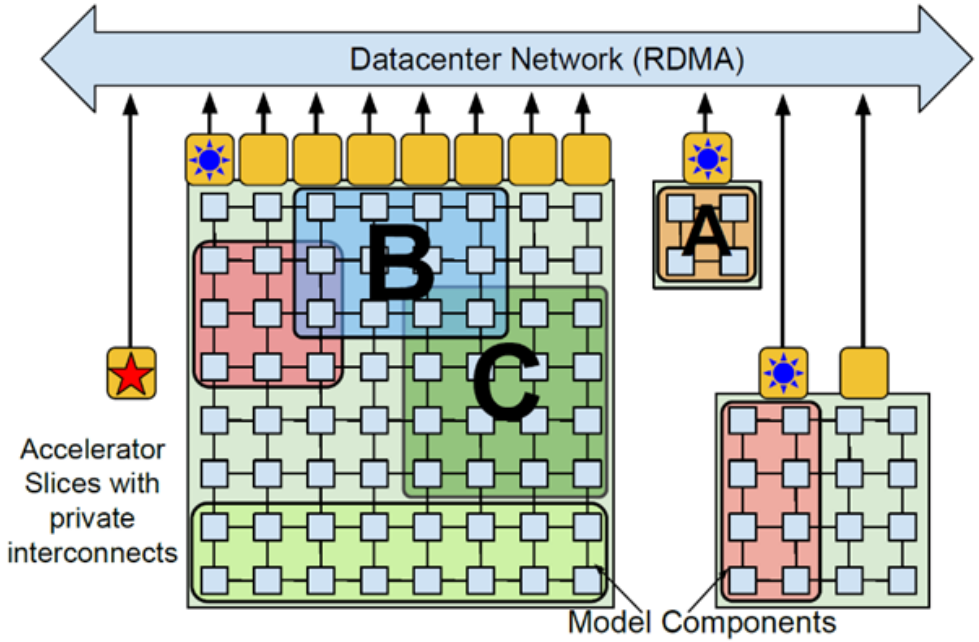
Is Parallel Data Engineering of any Importance?

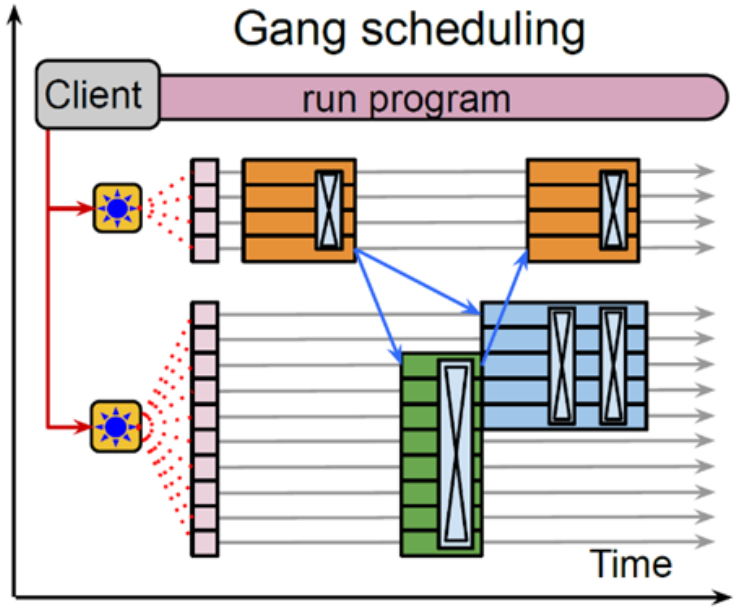
- **90%** of (future) **AI for Science** is **Deep Learning**
- In a DL job perhaps **10% of user code** but **95% of execution time** is in Tensorflow or PyTorch; rest is data engineering
- So we need to both use parallel TensorFlow and PyTorch (understood) and integrate all the parallel components as likely to have many AI jobs
  - Often 512 GPUs maximum useful size so AI supercomputer will simultaneously run lots of jobs
- Note Deep Learning unusually sensitive to I/O
  - Read and Write Models
  - Read and Shuffle Data
- Must re-use existing technologies: PyTorch JAX Pandas Workflow (Ray, HPC)
- Use best Industry and Research technologies
- MLIR very attractive as used commercially to both optimize basic code (e.g. linear algebra OpenXLA) and describe systems
- Make Pathways open source …….

Data Science/Data Engineering Execution Model

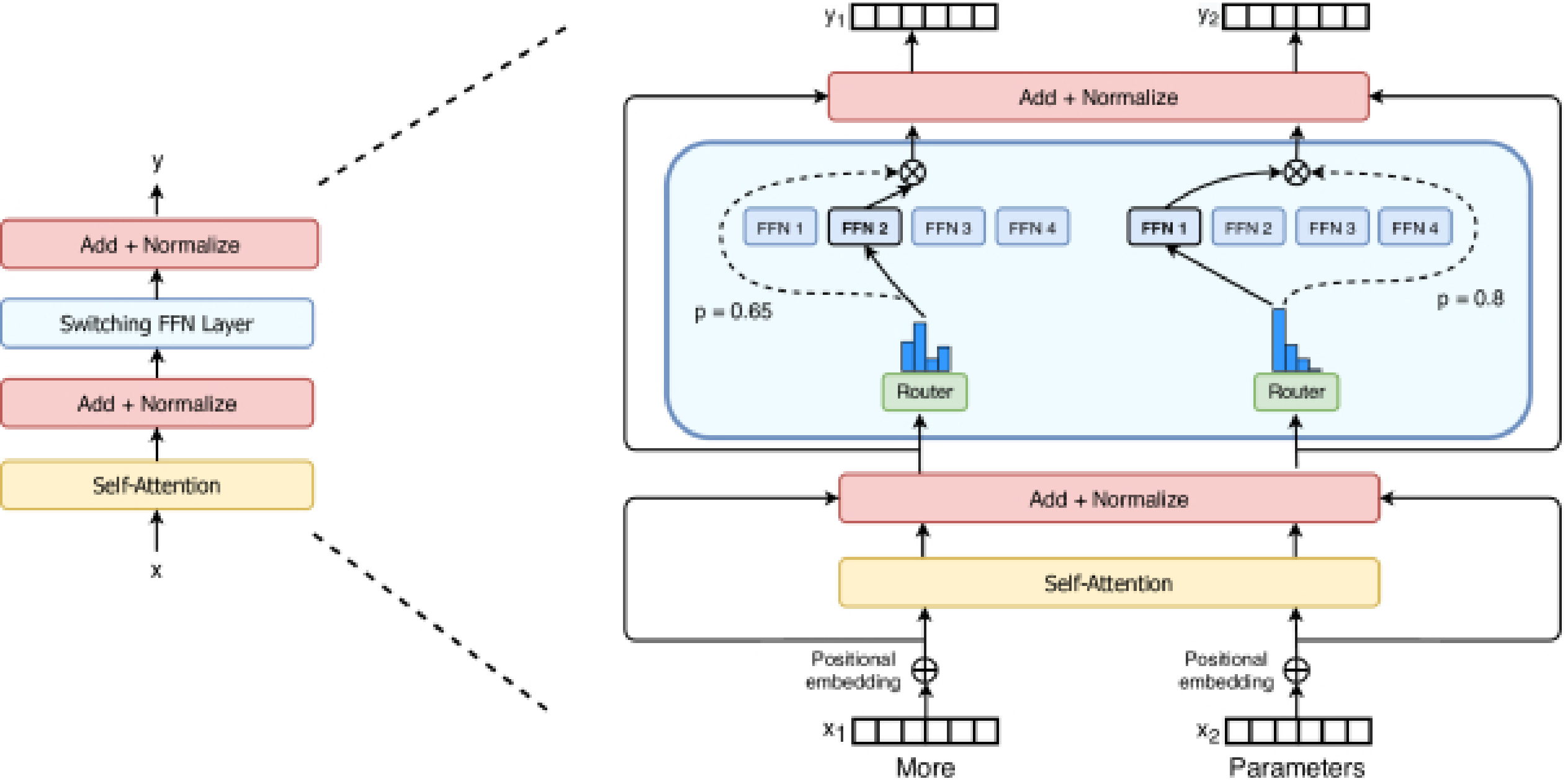
- Deep Learning needs an efficient mixture of asynchronously scheduled tasks linked by dataflow where each task is classic gang scheduled Bulk Synchronous Processed
- Hyperparameter search is asynchronous as are parts of a deep learning model
- Picture is from Google Pathways that implements this model for deep learning
- Naturally extended to a mix of deep learning (machine learning) and data engineering







Expert Parallelism in a Switch Transformer



Unbundling Deep Learning with Operators

- Typically switch to black-box PyTorch or Tensorflow for deep learning with limited opportunities for custom optimizations for compute and I/O
- Google Pathways paper describes an “unbundled operator” approach to deep learning that could use “conventional HPC tools” to implement complete system
  - Shuffling
  - Batching
  - Disk--CPU--GPU memory management
  - Vector (Numpy) to Tensor adding metadata to enable differentiation
  - Forward Loss calculation pass
  - Back propagation pass (JAX)
  - Save Model
  - Conventional operations in Pandas/Numpy/Cylon
  - Workflow scheduling
  - Other DNN capabilities such as nonlinear neurons and second order optimization
- Could add Compiler-based automation for operator parallelism
- Need open-source framework to enable uniform treatment of deep learning and data engineering

Some deep learning operators

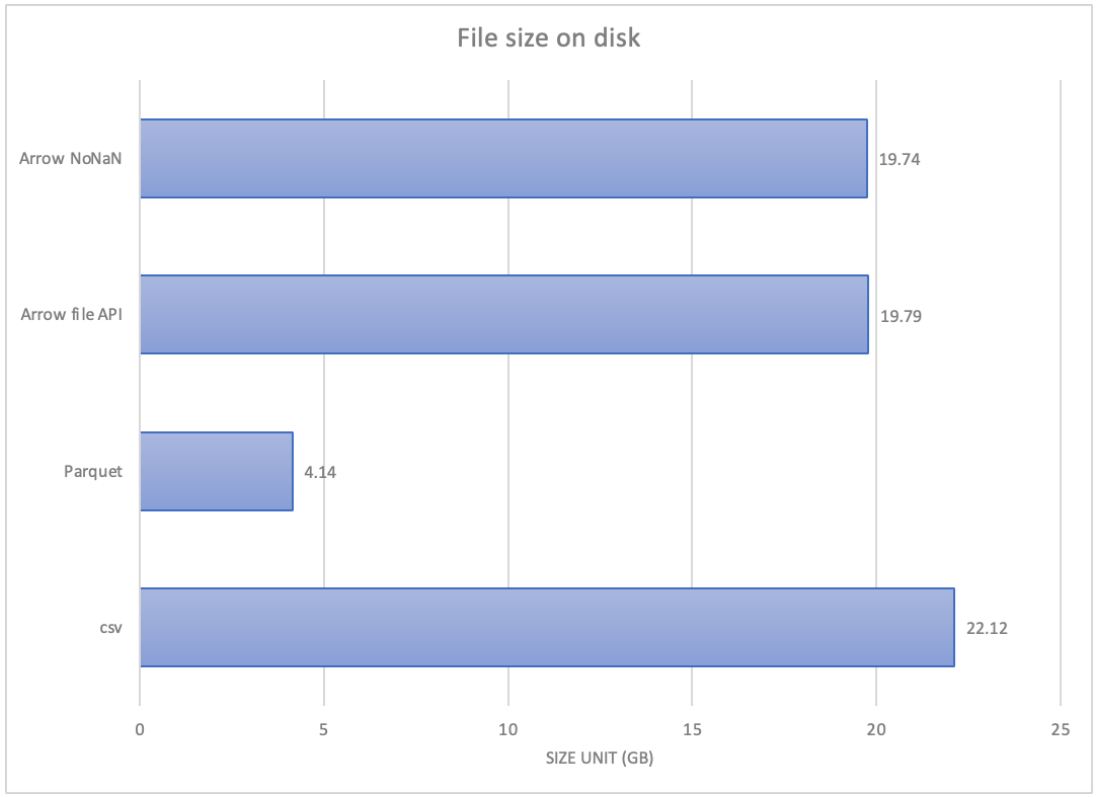
Systems Approach

- **Pathways** implemented with **Google PLAQUE** workflow environment but suggests would run on Ray distributed OS
- Should be able to use **HPC workflows** such as Radical Pilot, Pegasus, Parsl
- Add full **Data Engineering to Deep Learning** with complete set of data engineering and deep learning operators
- Support many simultaneous DE+DL jobs
- Specify program in **MLIR** using feature that MLIR specifies a general **dataflow graph** of tasks
- **Specify hardware system** -- CPU GPU Disks and Networks -- perhaps also in MLIR
  - Need a **distributed system Ontology** for this and for FAIR metadata of benchmarks and any data involving computer systems
- Map program to hardware ("runtime compiler")
  - Pathways static but applications need **dynamic scheduling**
  - Choose **operator parallelism** to schedule on available "MPI Islands"
  - **Scheduling using Reinforcement Learning** probably best
- Support **Arrow and Parquet for parallel I/O**

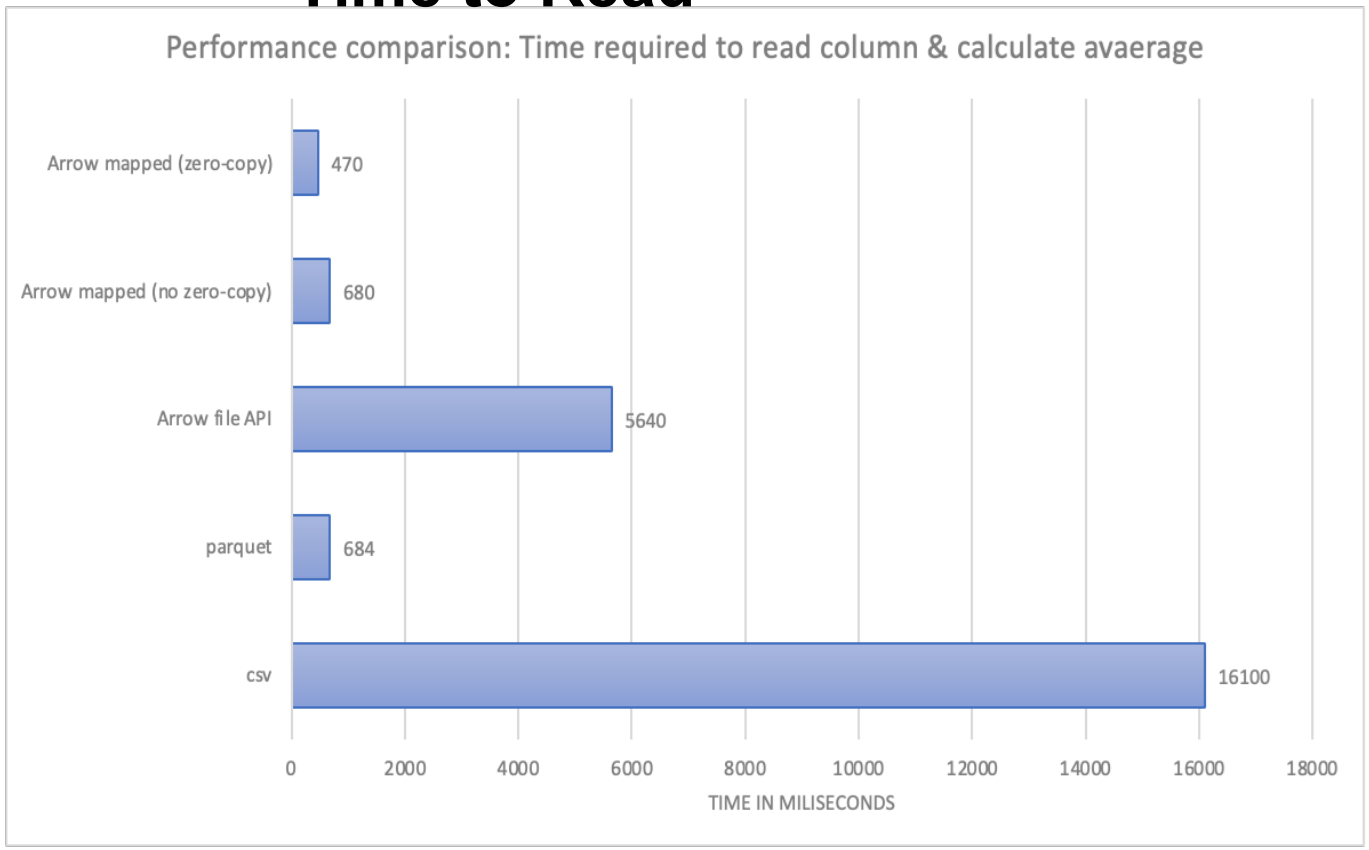
CSV vs. Parquet vs. Arrow file

Tianle Zhong
Parquet high performance and compact disk storage

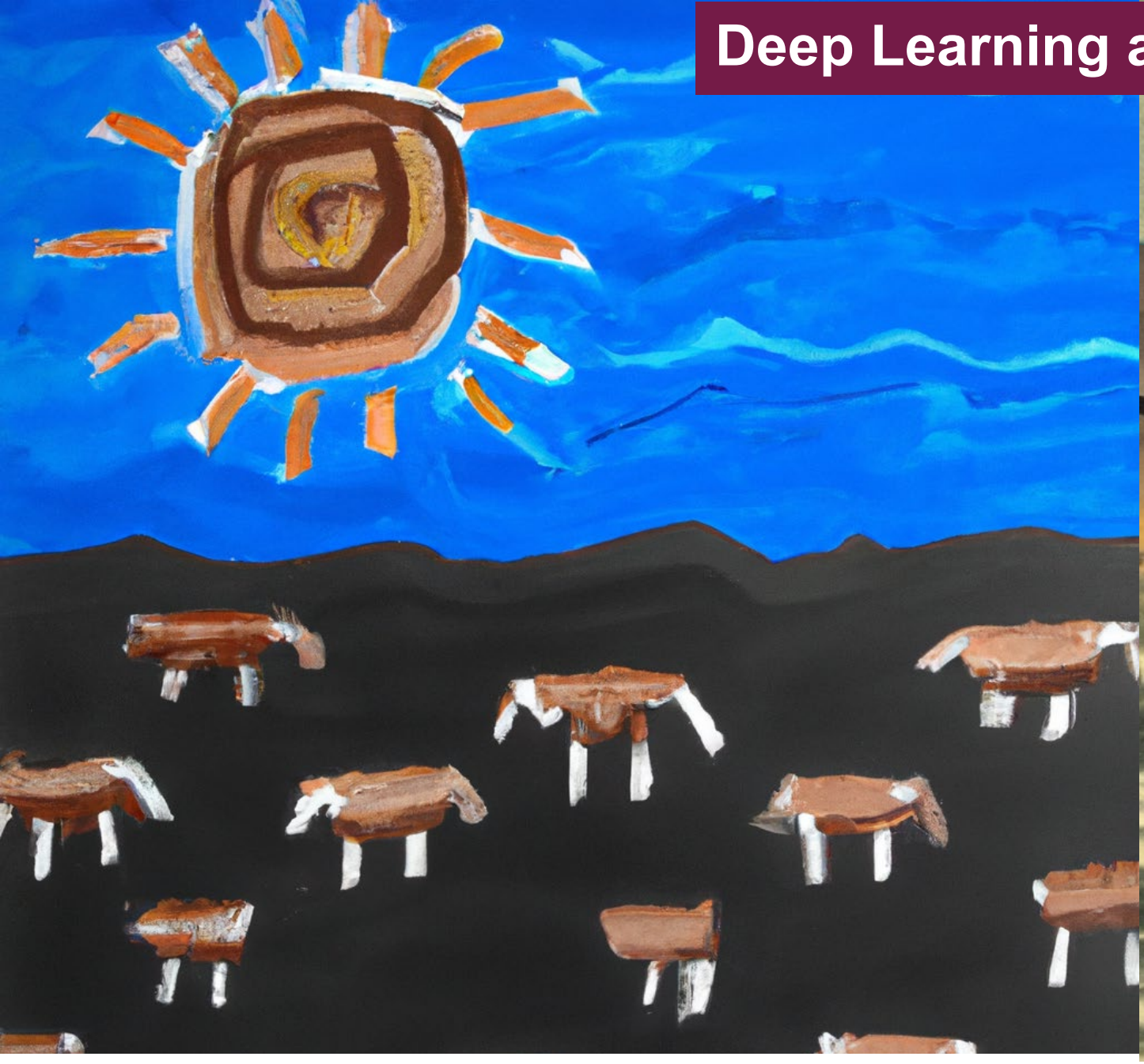
Size on Disk



Time to Read



Deep Learning and Data Engineering



DALL-E: “An ugly painting of the sun shining on a stormy sea with ten small black robots swimming in the sea in style of an amateur”

Stable Diffusion

Conclusion

Conclusions

- **AI** promises pervasive transformative approach to **enhance Science Discovery**
  - Deep Learning transforming simulations (surrogates), data analytics(data from simulations or observation), apparatus (AI will design, monitor, control experiments), reasoning
- Need to develop **meta-models** (**Foundation** models) valid across many domains
  - At least understand how to move across domains and common issues
  - Broad use of deep and reinforcement learning
- **HPC is essential** for AI
- **Data Engineering** can be parallelized via a library of parallel operators from database to Pandas style data transformations
- **Deep Learning** has growing need for more flexible parallelization
- Efficient integration of **Deep Learning** and **Data Engineering** is likely to work well but challenging due to multiple languages and multiple distributed environments
  - Use of MLIR to describe systems of tasks and hardware promising

DALL-E

‘’A cute fox lives in a house made out of sushi”

Questions?