Deep RC: A Scalable Data Engineering and Deep Learning Pipeline

Arup Kumar Sarker*, Aymen Alsaadi, Alexander James Halpern, Prabhath Tangella, Mikhail Titov, Niranda Perera, Mills Staylor, Gregor von Laszewski, Shantenu Jha, Geoffrey Fox

University of Virginia*, Biocomplexity Institute and Initiative, Princeton University, Princeton Plasma Physics Laboratory, Rutgers University, Brookhaven National Laboratory, Nvidia Corporation

June 4, 2025

Overview

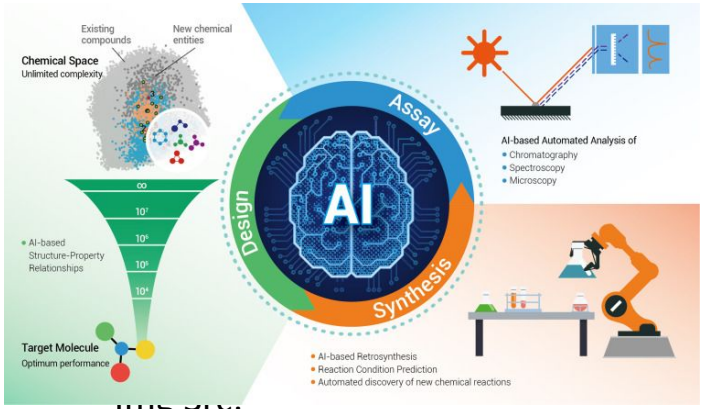
IPDPS
Milano, Italy
June 3-7, 2025

- Objective and Motivation
- Prior Arts
- Research Problems
- Proposed Research Solutions
- DeepRC Architecture
- Evaluations
- Conclusions and Future Works

Objective and Motivation

IPDPS
Milano, Italy
June 3-7, 2025

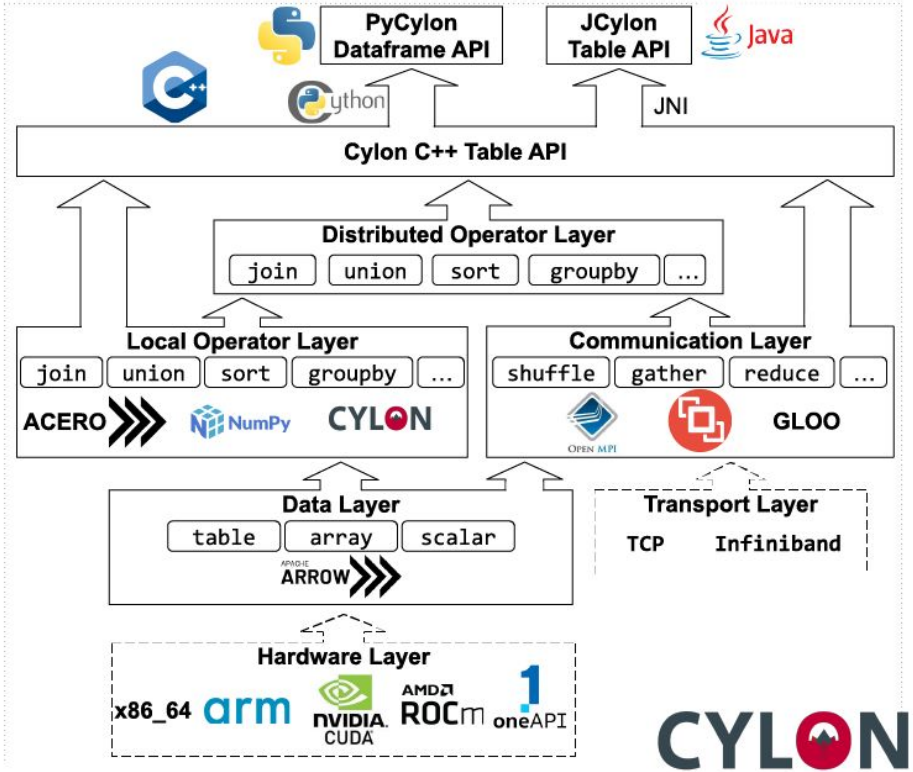
- The use of Deep Learning in Scientific computing is increased with data volume and complexity massively.
- Massive investment in cloud and relevant frameworks solves new problems and the total computing power in the world has quadrupled due to AI engine deployment
- Big Data on top of AI and ML infrastructure, along with clouds, is the new direction.
- Heterogeneity, high dimensionality, and complex relationships between variables.
- An efficient scalable system is the key to handling these challenges.



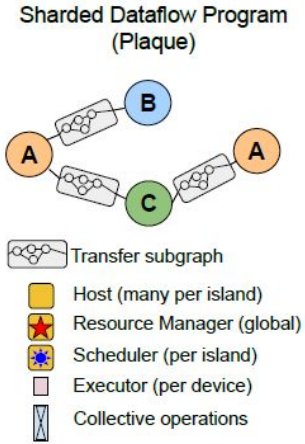
https://www.sciencedirect.com/science/article/pii/S2666675821001041

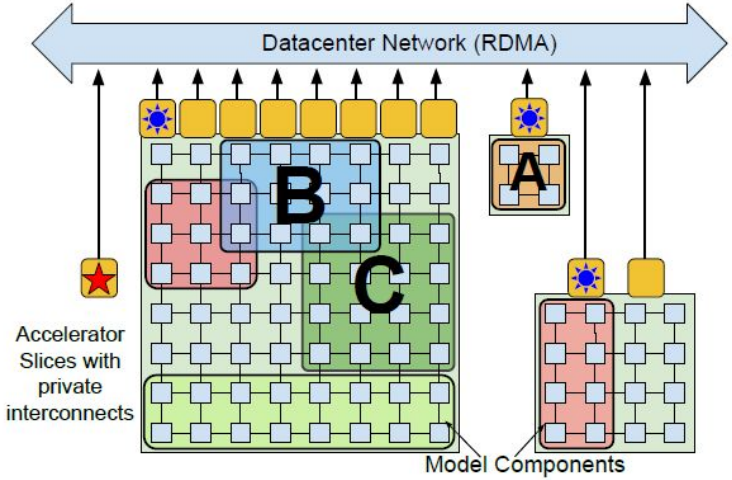
Prior Art: Cylon

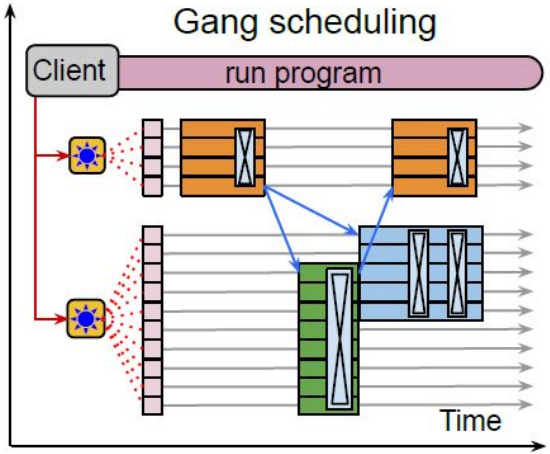
- A distributed memory data-parallel dataframe is used to meet objectives.
- It utilizes the Bulk Synchronous Parallel (BSP) execution model and adopts SPMD pattern.
- Arrow columnar data is used with procedural abstractions through dataframe operators and communication operators.
- It comprises data structures such as dataframes/tables, columns, and scalars.
- Cylon has to use batch execution which is inefficient for heterogeneous data pipeline.



Prior Art: PATHWAYS



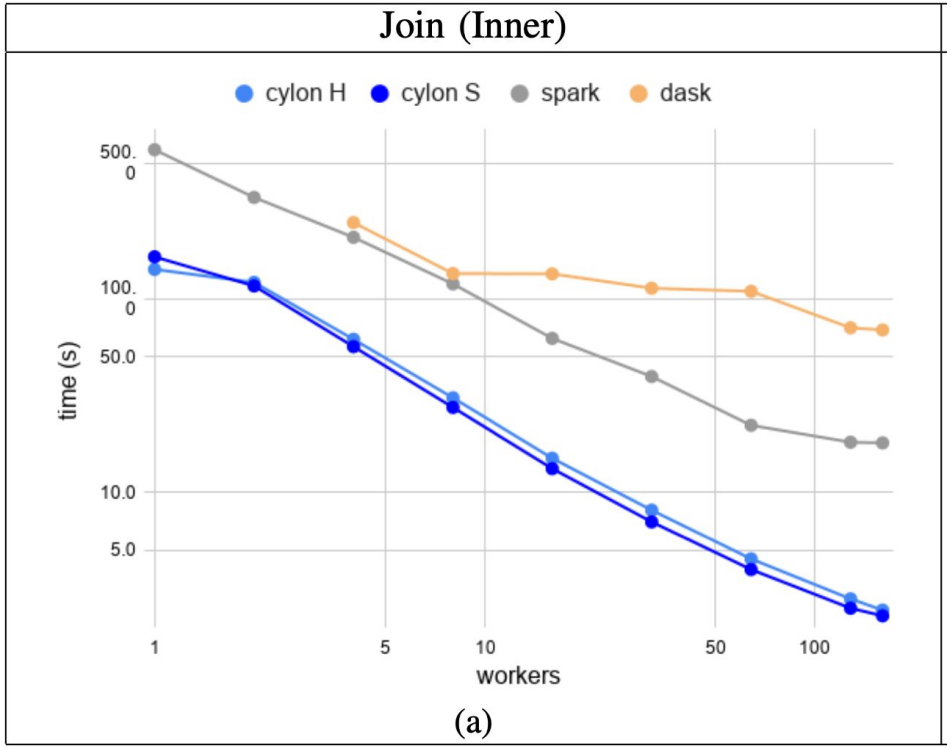




- Pathways points the way to the direction of high-performance computing.
- It provides a sophisticated execution environment for deep learning jobs.
- Supports gang-scheduled dynamic and parallel asynchronous dispatch with conflict resolution on interdependent resource/output sharing.
- However, it is closed source and only compatible with Google core infrastructures.

Prior Art: DASK, SPARK

- DASK and Spark Dataframe can be alternatives to CYLON, but CYLON Data frames outperforms both in multiple scaling operations.

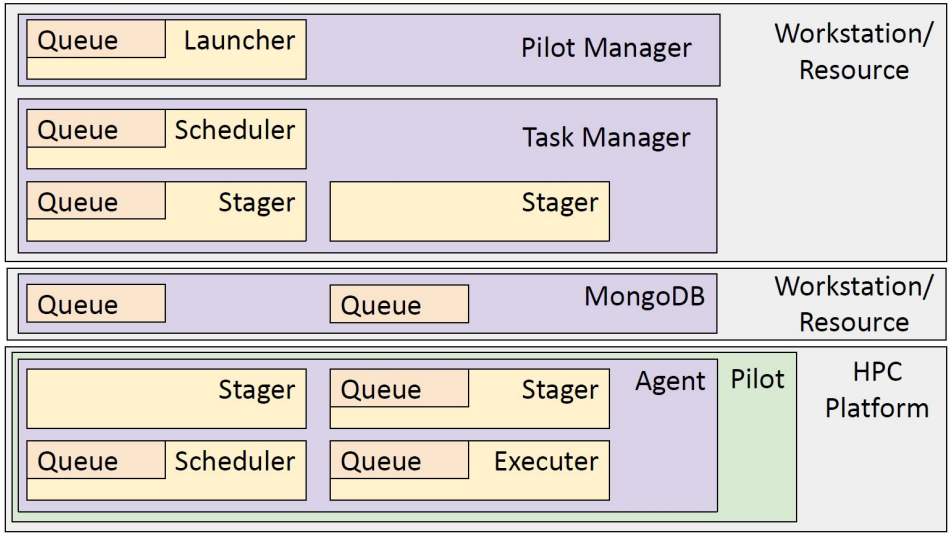


Prior Art: RAPIDS, RAY

- RAPIDS-cuDF is highly optimized for NVIDIA GPUs. GPU Cylon is in the pipeline for scaling operations and Radical-Cylon is designed to support GPUs.
- RAY provides a distributed runtime that can be an alternative to radical-pilot as workflow engine. Cylon Integration with Ray will overcharge the distributed processing. We do have plans to work Cylon on Rays in Future Works.
- Ray's GPU support is restricted to scheduling and reservations where Radical pilot has the flexibility to control underlying resources.

Prior Art: Radical-Cylon

- Radical function provides heterogeneous execution with multiple data pipelines.
- The design supports heterogeneous resources (CPUs, GPUs).
- Cylon join and sort distributed operations can be executed in two different pipelines.
- It solves the use batch execution which is inefficient for heterogeneous data pipeline.
- Data (pre/post) processing can be implemented by reusing Radical Pilot for workflow and Cylon for dataframes.
- But Radical-Cylon does not have Deep Learning pipeline.



Research Problems

- Cylon faces limitations in solving the challenge of multiple data pipeline execution.
- It requires an underlying execution environment, and batch operations are performed as executables, hindering compatibility with heterogeneous execution.
- For instance, joining in one pipeline and ML inference in another is not supported. The need for an underlying task-based execution environment is apparent.
- Resource management is predetermined, relying on SLURM-based allocation, resulting in idle resources when a worker finishes a task, preventing their use for other pipelines in runtime.
- Lack of seamless execution of deep learning jobs.

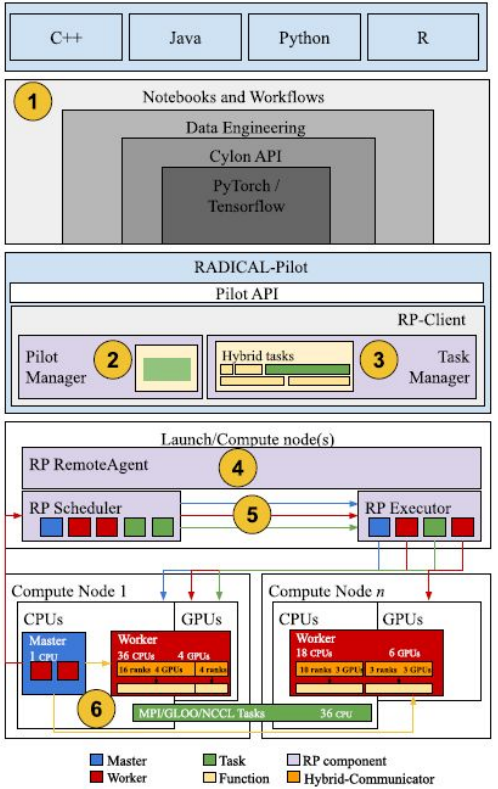
Proposed Research Solutions

- We propose Deep-RC, a task-based execution environment for Cylon where deep learning tasks are executed as Radical functions.
- Resources are allocated to multiple tasks, controlled by the RP Scheduler.
- We have introduced Deep RC bridge for seamless execution of data preprocessing and model inferencing jobs.
- To efficiently manage data loading in the Deep RC pipeline, the zero-copy data loader uses several workers to retrieve and preprocess data concurrently.
- This design allows for the efficient scheduling, placement, and launching of independent tasks across various compute nodes.
- It enables the execution of model inferencing jobs in separate pipelines.

Deep RC: A Scalable Data Engineering and Deep Learning Pipeline

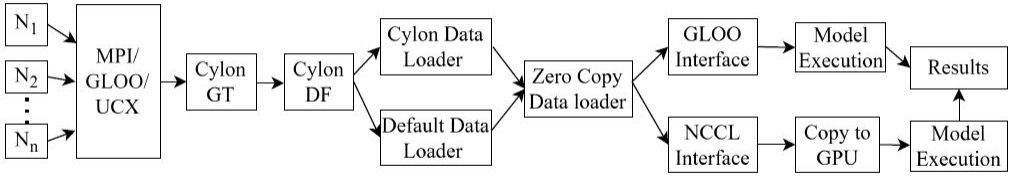
IPDPS
Milano, Italy
June 3-7, 2025

- Deep-RC is a scalable and modular runtime system that enables the efficient scheduling, placement, launching and execution of independent heterogeneous tasks across heterogeneous resources.
- It avoids the overheads of launching multiple MPI jobs by launching a single job and execute many MPI/GLOO/NCCL Python functions within.
- Cylon tasks take advantage of RP's capabilities by executing multiple multi-node MPI/GLOO/NCCL functions efficiently.
- DL training/prediction, Cylon join and sort distributed operation can be executed in two different pipeline.



Deep RC Bridge

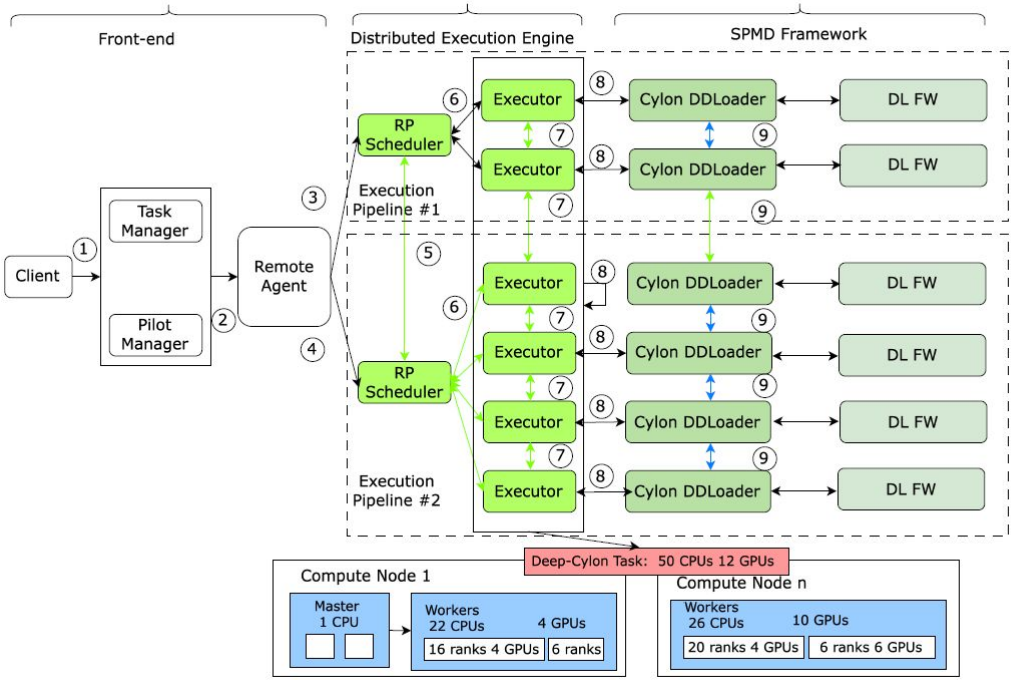
- Deep RC bridge, which allows data to be preprocessed using Cylon distributed data frames that operate on top of MPI/UCX/GLOO and produce a Cylon Global Table (GT).



- The global table, which may be zero-copied and translated to pandas and other data frame formats, is used to create the distributed Cylon dataframe.
- Before batches are loaded into memory, data transformations and augmentation are frequently implemented.
- The zero-copy data loader uses several workers to retrieve and preprocess data in simultaneously, effectively managing data loading in the Deep RC pipeline.
- It divides the workload among several subprocesses, each of which is in charge of loading a section of the dataset, rather than depending on a single process to load data sequentially. By preventing training bottlenecks, this parallelism makes sure the model gets data ASAP.

Deep RC Workflow

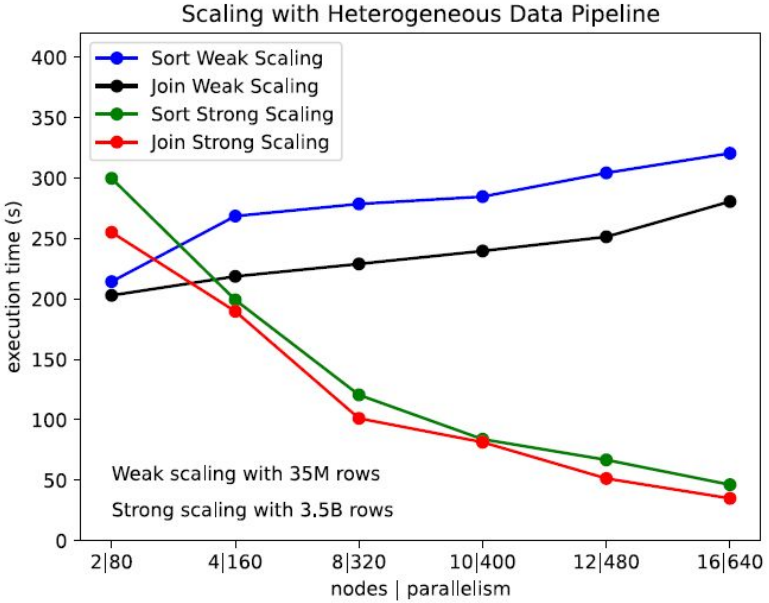
- Task Manager creates radical functions for all DL tasks
- Pilot manager allocates resources.
- Remote agent creates RP scheduler for multiple pipeline.
- Each scheduler resolves inter pipeline dependencies
- There are executors to handle each Deep Learning Jobs.
- Cylon dataframe will be used as a input for each Data Loader for model inference job.



Heterogeneous Execution: Weak and Strong Scaling

IPDPS
Milano, Italy
June 3-7, 2025

- For strong and weak scaling, Deep-RC archives expected behavior for heterogeneous data pipeline.
- Heterogeneous tasks (4 join/sort ws/ss) on a single execution.
- We use 35M rows for weak scaling and 3.5B rows for strong scaling with 40 cores/node.
- The scaling behavior of one data pipeline does not impact the other and shows expected scaling behavior.
- Resource allocation and releases did not create any circular dependencies.
- Increasing parallelism adds communication overheads; tasks are smaller.



Model Training/Prediction performance

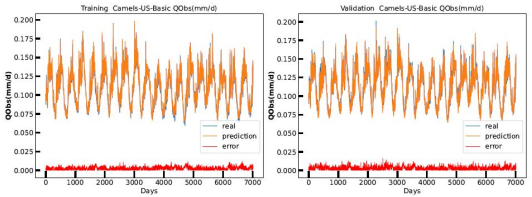


Fig. 8. Training(left) and Prediction(right) accuracy of Observed Streamflow(QObs) with Camels-US datasets in LSTM Hydrology model.

Table 3. Single pipeline testing by measuring Mean Absolute Error(MAE), Mean Squared Error(MSE), and Mean Absolute Percentage Error(MAPE) to ensure model training works as expected. Comparison of training time with Deep RC and Bare Metal Deep Learning(BM DL) with 400 epochs. We observe constant overheads(approximately 4.15 seconds on average) in a single pipeline.

Model	Train(s) BM DL	MAE	MSE	MAPE(%)	Train(s) Deep RC
Autoformer	185.51	0.51	0.57	2.65	189.15
DeepAR	72.14	0.50	0.59	2.48	76.32
NLinear	20.98	0.42	0.39	2.45	24.34
GRU	157.15	0.52	0.59	2.18	163.2
NBEATS	15.21	0.39	0.36	2.01	19.03
AutoNHITS	495.78	0.39	0.33	2.22	500.8
PatchTST	47.07	0.37	0.32	2.20	51.15
TFT	293.79	0.42	0.47	1.28	298.13
TimesNet	801.13	0.39	0.36	2.51	806.09
VanillaTransformer	264.57	0.46	0.51	1.43	269.18
TiDE	19.39	0.36	0.34	1.84	23.61

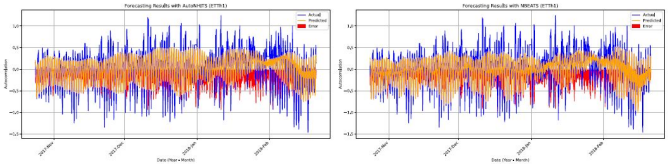


Fig. 9. Training and Prediction accuracy of electrical transformers' oil temperature for AutoNHITS and NBEATS model. The dataset used is the Electricity Transformer Dataset (ETDataset) [30]

- We got the expected precision in all models and generated the results with 3 metrics (MAPE, MAE, MSE) shown in Table-3.
- Deep RC smoothly supports both PyTorch and Tensorflow, we test the Tensorflow pipeline using the Hydrology Model and the PyTorch pipeline using 11 models from Neuralforecast .
- Although the total execution times of the Deep RC and BM Deep Learning approaches differ by 1 to 5 seconds, training time is less and does not impact on prediction process.
- Aside from the comparable performance, we observe a constant overhead when using Deep RC in strong scaling operations despite increasing parallelism.

Model Training/Prediction performance

IPDPS
Milano, Italy
June 3-7, 2025

- We see constant overheads between 6 to 8 seconds compared to a total time of 6482.24 to 14456.64 seconds which is very negligible.
- We have developed 11 pipelines with one Cylon join and 11 deep learning inferencing jobs using an LSTM-based Hydrology and NeuralForecast model.
- It significantly decreased 75.9 and 3.28 seconds in both experiments, which is important for inferencing tasks.
- Because any commercial cloud platform that manages thousands of requests would be significantly impacted.

Table 2. Deep RC Execution Time and RP Overheads of different Neural Forecast and Hydrology models with Training Task on Rivanna.

Forecasting Domain	Model	GPUs a100 80GB	Execution Time time (seconds)	Overheads (tasks/second)
Hydrology	LSTM	1	14456.64 ±4.97	4.13 ±1.1
		2	10216.52 ±4.26	4.13 ±1.6
		4	6482.24 ±5.13	5.01 ±1.82
Neural Forecast	Autoformer	2	189.15 ±3.23	3.56 ±0.8
	AutoNHITS	2	500.8 ±3.13	3.51 ±0.41
	TFT	2	298.13 ±3.23	3.12 ±0.9
	TimesNet	2	806.09 ±4.84	4.69 ±0.21
	DeepAR	2	72.32 ±3.35	3.45 ±0.51
	VanillaTransformer	2	269.18 ±4.67	4.56 ±0.33

Table 4. Performance Comparison of multiple pipelines with Deep RC and Bare Metal Deep Learning(BM DL) with PyTorch and TensorFlow based models, runs on 2 a100s 80GB GPUs. For cylon job, it uses 8 nodes with 40 cores/node

Pipeline Type	Number of pipelines	Cylon Task	DL Task	BM-DL Execution(s)	Deep RC Execution(s)
Hydrology	11	join	inferencing	21381.73135	21305.83772
NeuralForecast	11	join	inferencing	167.8454124	164.5677196

Limitations and Future Works

- However, when we attempted to infer LLMs, the RP scheduler and resource allocation module—had trouble assigning resources for Deep RC.
- To manage such jobs, a design modification incorporating multi-level parallelism is necessary. This significant design modification will be presented separately because we view it as a future work.
- The unified execution model depends on the Arrow engine, and there are multiple scopes to optimize Arrow-based data queries.
- We plan to support all multi-tenancy requirements, e.g. prioritization, performance isolation, and resource tracking in the future.

Conclusions

- Deep RC offers an unified framework that simplifies model training, prediction, and data processing under a centralized execution paradigm by tackling the challenges of resource management and varied pipeline execution.
- The proposed design tackles the intricacies of resource management and significantly decreased 75.9 and 3.28 seconds in both experiments, which is important for inferencing tasks.
- This transformation introduces an efficient resource management and scheduling framework that interfaces between the client and cluster nodes.
- Deep RC's versatility across a range of distributed tasks, including the capacity to smoothly interleave client workloads, optimize deep learning execution pipelines, and reduce computational overhead.

References

- Sarker, A.K., Alsaadi, A., Perera, N., Staylor, M., von Laszewski, G., Turilli, M., Kilic, O.O., Titov, M., Merzky, A., Jha, S., et al.: Radical-cylon: A heterogeneous data pipeline for scientific computing. In: Workshop on Job Scheduling Strategies for Parallel Processing. pp. 84–102. Springer (2024)
- Niranda Perera, Kaiying Shan, Supun Kamburugamuwe, Thejaka Amila Kanewela, Chathura Widanage, Arup Sarker, Mills Staylor, Tianle Zhong, Vibhatha Abeykoon, Geoffrey Fox, "*Supercharging Distributed Computing Environments For High Performance Data Engineering*, " https://arxiv.org/pdf/2301.07896.pdf
- Abeykoon, Vibhatha and Kamburugamuve, Supun and Widanage, Chathura and Perera, Niranda and Uyar, Ahmet and Kanewala, Thejaka Amila and von Laszewski, Gregor and Fox, Geoffrey, "*HPTMT Parallel Operators for High Performance Data Science and Data Engineering*" https://doi.org/10.3389/fdata.2021.756041
- Widanage, Chathura and Perera, Niranda and Abeykoon, Vibhatha and Kamburugamuve, Supun and Kanewala, Thejaka Amila and Maithree, Hasara and Wickramasinghe, Pulasthi and Uyar, Ahmet and Gunduz, Gurhan and Fox, Geoffrey, "*High performance data engineering everywhere*", https://arxiv.org/pdf/2007.09589.pdf
- Kaiying Shan, Niranda Perera, Damitha Lenadora, Tianle Zhong, Arup Kumar Sarker, Supun Kamburugamuve, Thejaka Amila Kanewela, Chathura Widanage, and Geoffrey Fox. 2022. Hybrid Cloud and HPC Approach to High-Performance Dataframes. In 2022 IEEE International Conference on Big Data (Big Data). IEEE, 2728–2736.
- A. Merzky, M. Turilli, M. Titov, A. Al-Saadi, and S. Jha. 2022. Design and Performance Characterization of RADICAL-Pilot on Leadership-Class Platforms. IEEE Transactions on Parallel and amp; Distributed Systems 33, 04 (apr 2022), 818–829. https://doi.org/10.1109/TPDS.2021.3105994
- Paul Barham, Aakanksha Chowdhery, Jeff Dean, Sanjay Ghemawat, Steven Hand, Daniel Hurt, Michael Isard, Hyeontaek Lim, Ruoming Pang, Sudip Roy, et al. 2022. Pathways: Asynchronous distributed dataflow for ML. Proceedings of Machine Learning and Systems 4 (2022), 430–449.
- Jeffrey Dean and Sanjay Ghemawat.2008. MapReduce:simplified data processing on large clusters. Commun. ACM 51, 1 (2008), 107–113.

References

- A. McKenna. 2010. The Genome Analysis Toolkit: a MapReduce framework for analyzing next-generation DNA sequencing data. Genome research 20 9 (2010), 1297–303. https://doi.org/10.1101/gr.107524.110
- Wes McKinney et al. 2011. pandas: a foundational Python library for data analysis and statistics. Python for high performance and scientific computing 14, 9 (2011), 1–9.
- Philipp Moritz, Robert Nishihara, Stephanie Wang, Alexey Tumanov, Richard Liaw, Eric Liang, Melih Elibol, Zongheng Yang, William Paul, Michael I Jordan, et al. 2018. Ray: A distributed framework for emerging {AI} applications. In 13th USENIX Symposium on Operating Systems Design and Implementation (OSDI 18). 561–577.
- NirandaPerera,ArupKumarSarker,MillsStaylor,GregorvonLaszewski,Kaiy- ing Shan, Supun Kamburugamuve, Chathura Widanage, Vibhatha Abeykoon, Thejaka Amila Kanewela, and Geoffrey Fox. 2023. In-depth analysis on parallel processing patterns for high-performance Dataframes. Future Generation Computer Systems (2023).
- Pavel Shamis, Manjunath Gorentla Venkata, M Graham Lopez, Matthew B Baker, Oscar Hernandez, Yossi Itigin, Mike Dubman, Gilad Shainer, Richard L Graham, Liran Liss, et al. 2015. UCX: an open source framework for HPC network APIs and beyond. In 2015 IEEE 23rd Annual Symposium on High-Performance Interconnects. IEEE, 40–43.
- Jinhui Yuan, Xinqi Li, Cheng Cheng, Juncheng Liu, Ran Guo, Shenghang Cai, Chi Yao, Fei Yang, Xiaodong Yi, Chuan Wu, et al. 2021. Oneflow: Redesign the distributed deep learning framework from scratch. arXiv preprint arXiv:2110.15032 (2021).
- Matei Zaharia, Reynold S Xin, Patrick Wendell, Tathagata Das, Michael Armbrust, Ankur Dave, Xiangrui Meng, Josh Rosen, Shivaram Venkataraman, Michael J Franklin, et al. 2016. Apache spark: a unified engine for big data processing. Commun. ACM 59, 11 (2016), 56–65.
- Jeff Dean. 2021. Introducing Pathways: A next-generation AI architec- ture. https://blog.google/technology/ai/introducing-pathways-next-generation- ai-architecture/. (Accessed on 04/17/2023).

Thank you!

djy8hg@virginia.edu