The State of XGBoost: history and community overview

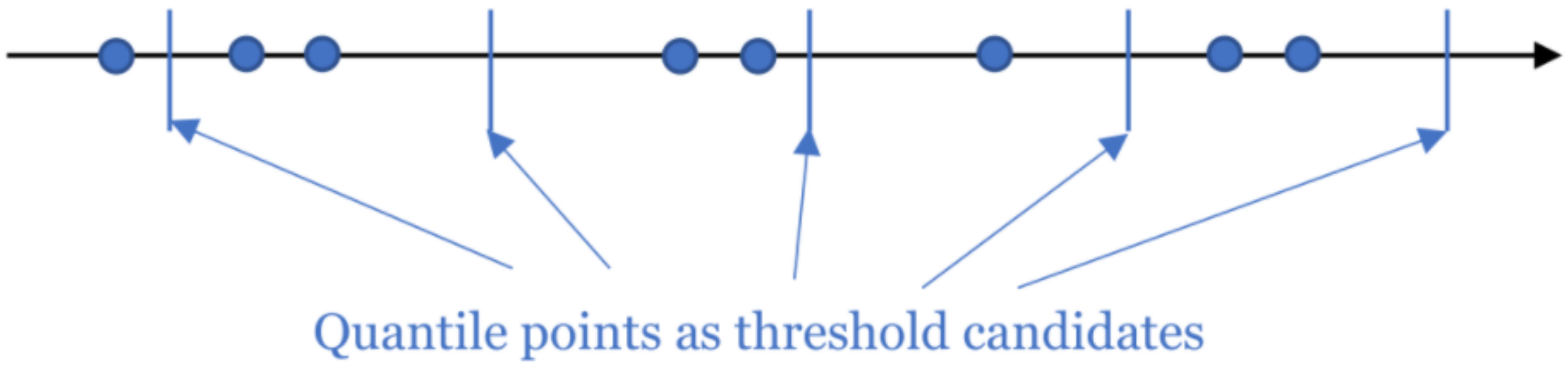
Hyunsu Cho
NVIDIA, RAPIDS AI
November 10, 2020

History of XGBoost

- Feb 2014: Initial commit
- Sept 2014: Submission R package to CRAN
- Aug 2015: First submission to PyPI
- May 2015: Scikit-learn integration
- July 2016: Spark integration
- Aug 2016: Publication in ACM KDD

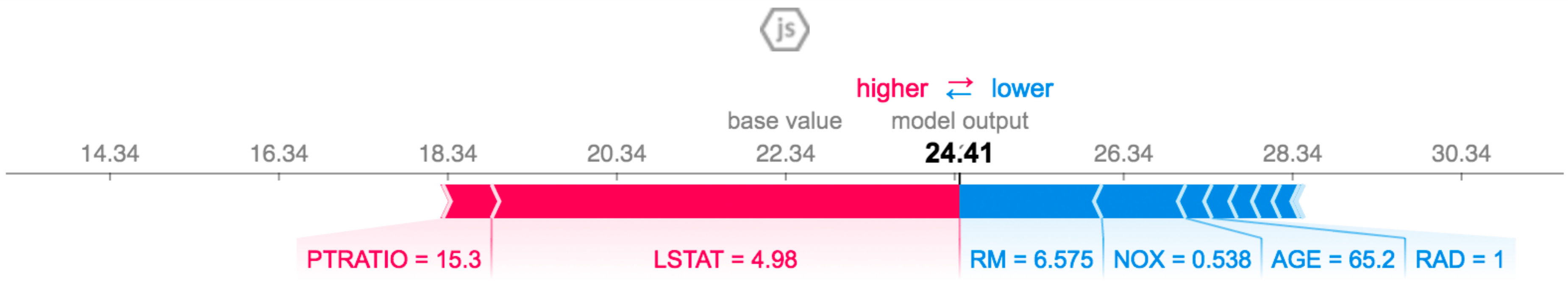
History of XGBoost

- Dec 2017: Experimental support for NVIDIA GPUs
- Dec 2017: Faster tree construction algorithm ('hist') with approximate quantiles



History of XGBoost

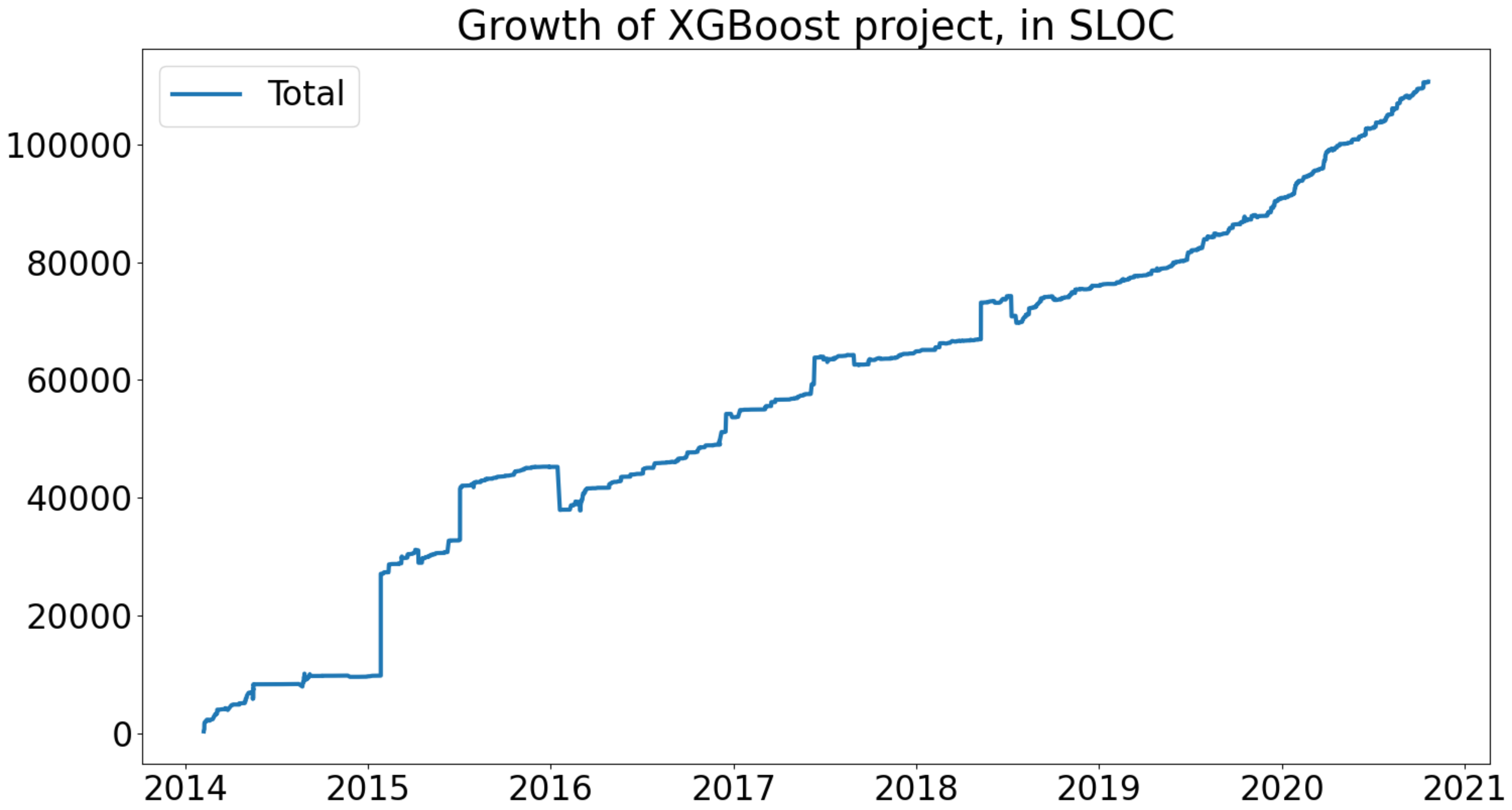
- Dec 2017: Experimental support for NVIDIA GPUs
- Dec 2017: Faster tree construction algorithm ('hist') with approximate quantiles
- Dec 2017: Feature contribution with SHAP

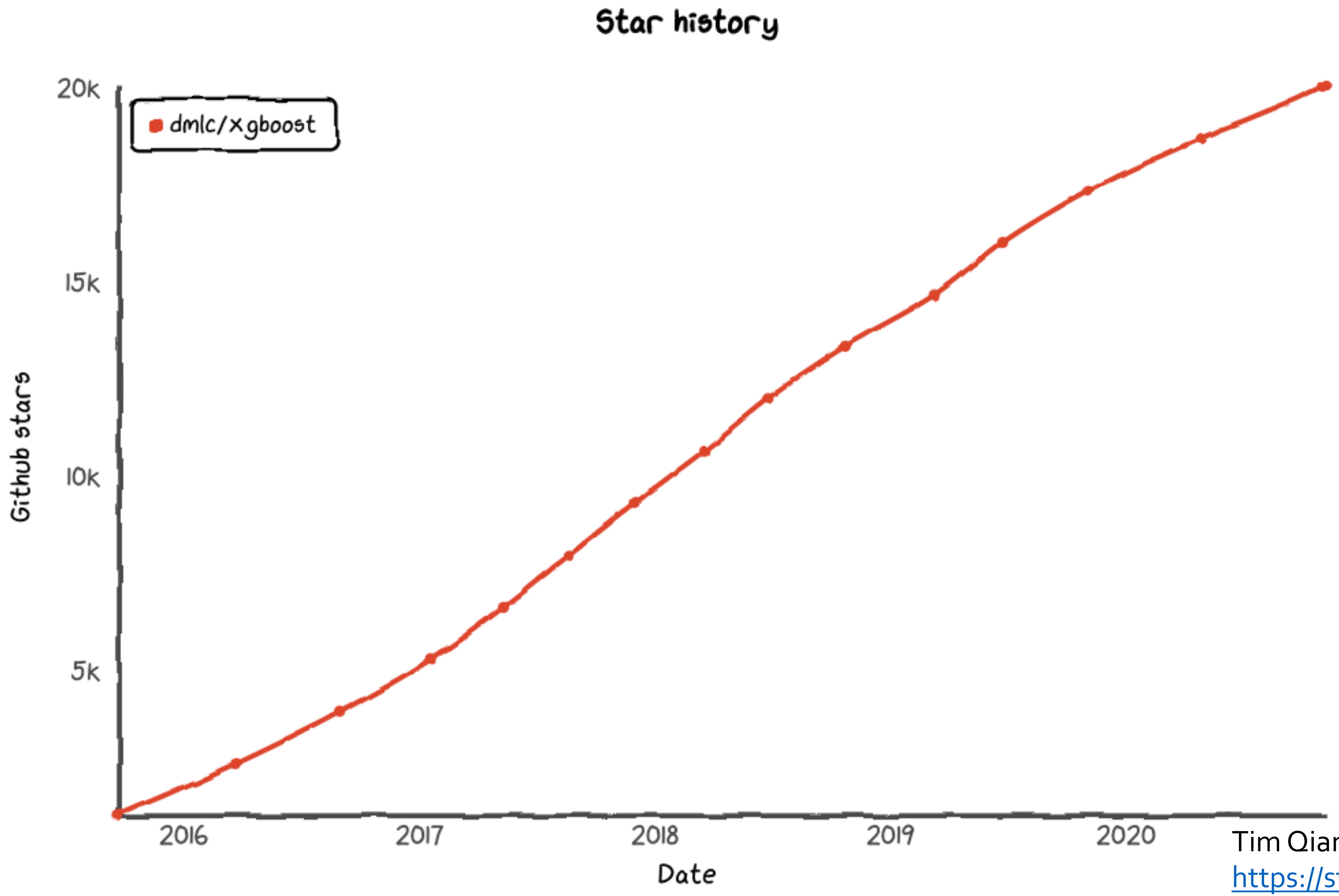


Scott M. Lundberg, Su-In Lee,
"A Unified Approach to Interpreting Model Predictions," NIPS 2017

History of XGBoost

- Dec 2017: Experimental support for NVIDIA GPUs
- Dec 2017: Faster tree construction algorithm (‘hist’) with approximate quantiles
- Dec 2017: Feature contribution with SHAP
- June 2018: First-class support for NVIDIA GPUs
- Aug 2018: New Spark API in style of Mllib
- Jan 2019: Infoworld’s Technology of the Year Award
- Feb 2020: 1.0.0 release, Dask integration


Growth of XGBoost project, in SLOC
Total
100000
80000
60000
40000
20000
0
2014
2015
2016
2017
2018
2019
2020
2021

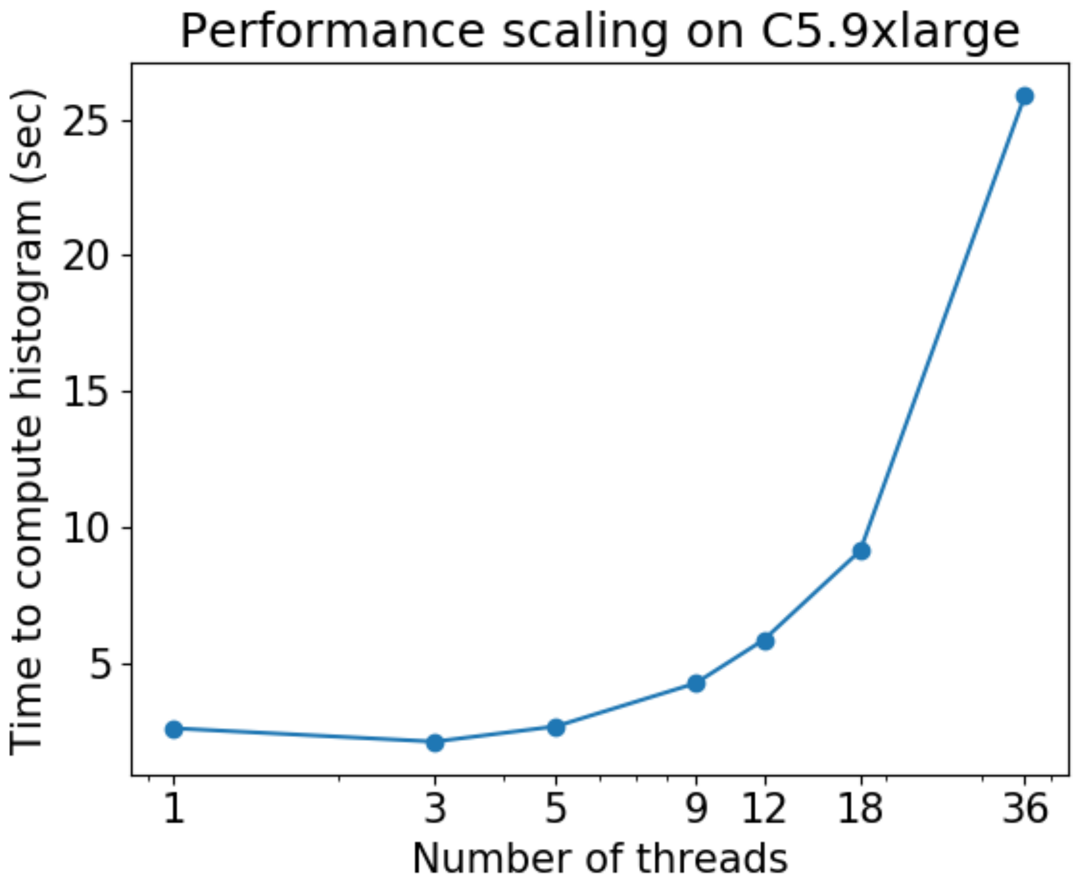


Tim Qian's star history
https://star-history.t9t.io/

Recent developments you should be excited about

Faster Performance on multicore CPUs

- Previously:



Faster Performance on multicore CPUs

- Intel contributed to improve performance scaling
  - Dec 2018 – Feb 2020
- Result shown in Szilard’s talk

New JSON model format

- Extensible
- Can be inspected by a human
- Easy to write parsers

Dask for Distributed Training

- Lightweight, easy to set up (pip / conda)
- Plays well with common data types, e.g. NumPy, Pandas
- Seamless interface for using **multiple NVIDIA GPUs** in single or multiple machines

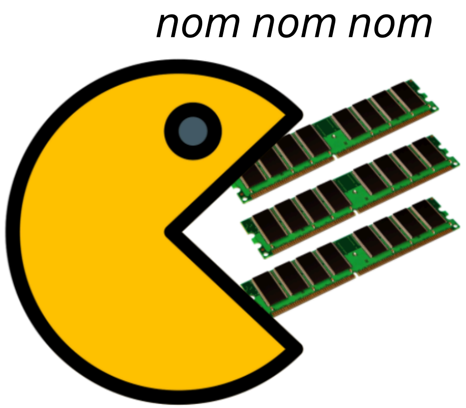
```
with LocalCluster(n_workers=4) as cluster:
    with Client(cluster) as client:
        bst = xgb.dask.train(client, ...)

with LocalCUDACluster(n_workers=4) as cluster:
    with Client(cluster) as client:
        bst = xgb.dask.train(client, ...)
```

Zero-copy data ingestion

- XGBoost is **memory-hungry**
- In Python, up to **three** copies of training data existed in memory!

(Intermediate format optimized for training)

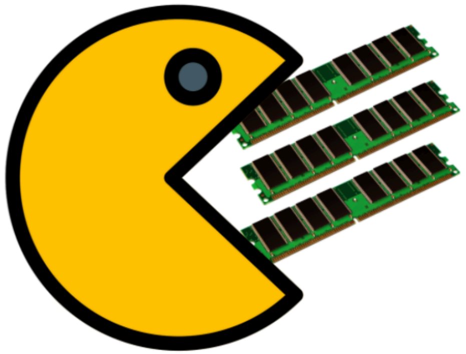


Zero-copy data ingestion

- XGBoost is **memory-hungry**
- In Python, up to **three** copies of training data existed in memory!
- Solution: consume data zero-copy / in-place.

  Currently implemented in GPU algorithm

nom nom nom

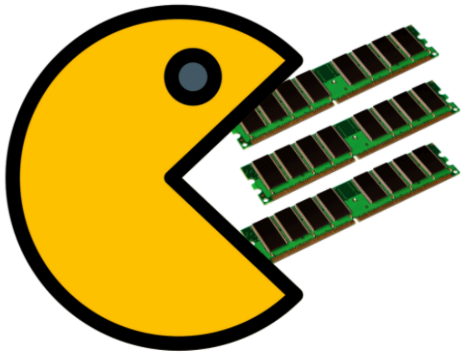


Zero-copy data ingestion

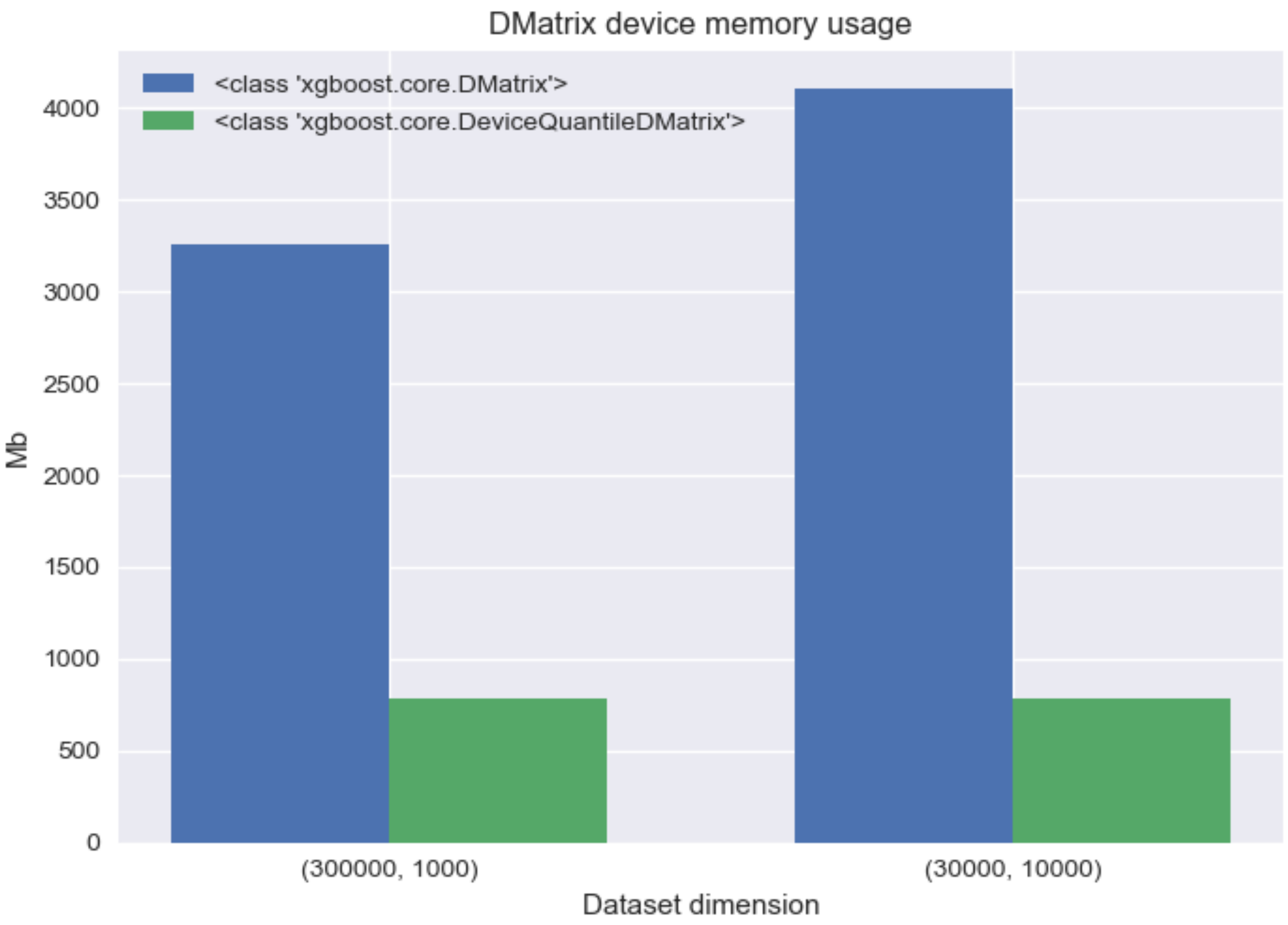
- XGBoost is **memory-hungry**
- In Python, up to **three** copies of training data existed in memory!
- Solution: consume data zero-copy / in-place.

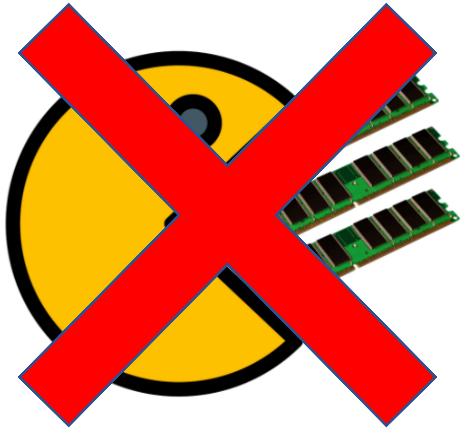
  Currently implemented in GPU algorithm
- **DeviceQuantileDMatrix**

nom nom nom



Zero-copy data ingestion





Zero-copy data ingestion

- XGBoost is **memory-hungry**
- In Python, up to **three** copies of training data existed in memory!
- Solution: consume data zero-copy / in-place.

  Currently implemented in GPU algorithm
- **DeviceQuantileDMatrix**
- **DaskDeviceQuantileDMatrix**: Zero-copy, with Dask arrays
  - Note: input should be GPU arrays already, e.g. cuPy

Zero-copy data ingestion

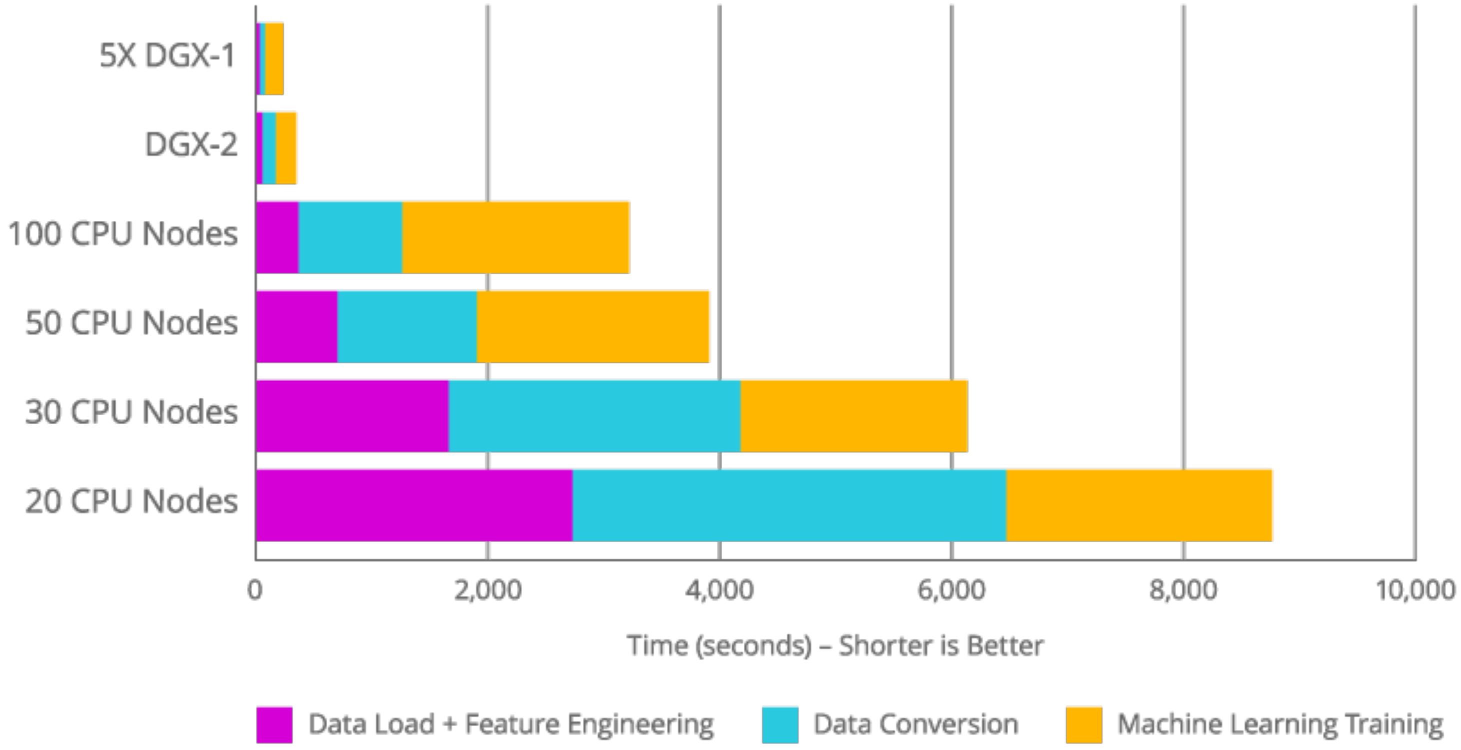
- XGBoost is **memory-hungry**
- In Python, up to **three** copies of training data existed in memory!
- Solution: consume data zero-copy / in-place.

In-place prediction: at prediction time, no need to build DMatrix

```
bst.inplace_predict(numpy_array)
bst.inplace_predict(cupy_array)
```

End-to-end Data Pipeline on GPU with RAPIDS

- Use NVIDIA GPUs to accelerate every part of data pipeline, including preprocessing, feature engineering
  - GPUs can even parse CSV files (!)



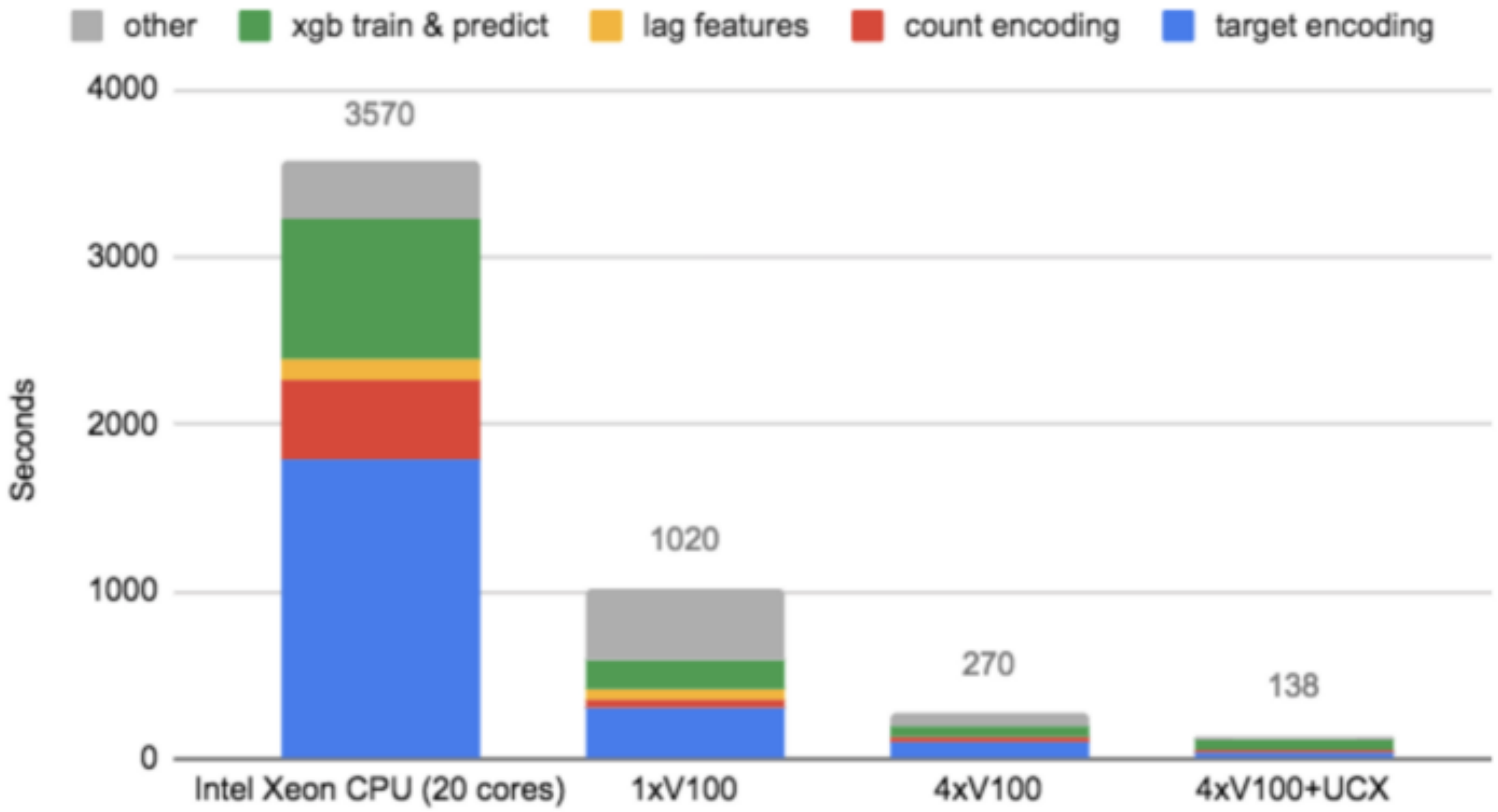
Iterate faster

Try lots of hyperparameters

"RAPIDS Accelerates Data Science End-to-End," https://developer.nvidia.com/blog/gpu-accelerated-analytics-rapids/

End-to-end Data Pipeline on GPU with RAPIDS

- First place in **Twitter RecSys Challenge 2020**



25x speedup over optimized CPU implementation

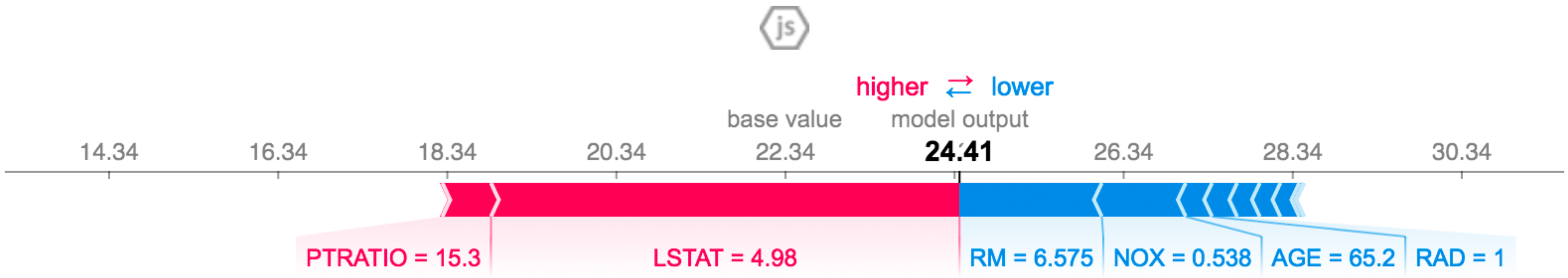
2 min. 18 sec. end-to-end

Benedikt Schifferer, Gilberto Titericz Junior, Chris Deotte, Christof Henkel, Kazuki Onodera, Jiwei Liu, Bojan Tunguz, Even Oldridge, Gabriel De Souza Pereira Moreira and Ahmet Erdem, "Accelerated Feature Engineering and Training for Recommender Systems." https://medium.com/rapids-ai/winning-solution-of-recsys2020-challenge-gpu-accelerated-feature-engineering-and-training-for-cd67c5a87b1f

GPU-Accelerated TreeSHAP

- Explaining feature contribution via SHAP is valuable but often slow
  - Especially pairwise feature interaction

- Use NVIDIA GPUs to accelerate SHAP
  Available in **Release 1.3.0** (est. end of month)



Rory Mitchell, Eibe Frank, Geoffrey Holmes, "GPUTreeShap: Fast Parallel Tree Interpretability," https://arxiv.org/abs/2010.13972

Improved testing

- Lots more tests for the C++ code base
- Automated testing farm validates all pull requests
- Make changes with confidence
- Make releases with confidence

Future Roadmap

- Categorical data support
- Share memory pool with other packages
- Clean up C++ codebase

- Does your business use XGBoost and would like to invest in major addition of capability?
- Consider dedicating developer(s) to improve XGBoost long term. E-mail phcho@nvidia.com if you're interested
- Many parts of XGBoost need care
  - R package
  - JVM packages
  - Swift/Ruby/Julia bindings
- Also consider making donations toward testing infrastructure