Scaling PyData with Dask & RAPIDS

Center for Machine Learning

Introduction

Kyle Nicholson

- Senior Software Engineer in Development Tools & Accelerators at the Center For Machine Learning
- Currently working on distributed and accelerated data science with Dask + Rapids
- Built and maintained a model logging library similar in function to MLFlow
- Dipping my toes into open source development
- Pursuing M.S. Computer Science at Georgia Tech
- B.S. Computer Engineering at Penn State

Introduction

C4ML

What are we going to cover today?

*The **challenges and potential to distribute and accelerate** financial and credit data analysis to build machine learning models, and how to **align an organization behind powerful open source tools** to **optimize value generation** across a large enterprise.*

Themes

Identifying the symptoms of lacking scalability

Utilizing OSS to deliver value faster

Leveraging GPUs for accelerated data science

Contributing to better your business & the community

A day in the life...
Data *Science* at Capital One

Data

C4ML

On an enterprise journey to ***deliver highly accurate business insights*** *by* ***consuming, processing, and analyzing*** *vast amounts of data faster*

Lots of Data + ***Drive for Faster Analysis*** = ***Need Distribution & Acceleration***

Very large data sets in a wide variety of formats

Data governance and federated access

Highly regulated environment to protect customer data

Large data science community to produce business insights

Enterprise initiatives to optimize data analysis

C4ML stood up to bolster enterprise ML capabilities

Need for simple, repeatable ways to stand-up large infrastructure

Huge interest in leveraging GPUs to accelerate compute

Many efforts to build custom solutions across the institution

This data landscape has created a large Data Science community at Capital One

Capital One

Data Science

C4ML

Large community of Data Scientists at Capital One with a wide variety of use cases, experience, skill sets, and programmatic preferences

Programming Language Prevalence

- A majority of data scientists utilize Python to get their jobs done
- A smaller subset of the community uses Java and Scala
- These languages often accompanied by Spark in GitHub repos

Scaling Python at Capital One

- Mostly by rewriting Python code to scale with Apache Spark
- Vertical scaling with very large memory and multi-core instances
- Custom solutions to scale Python for specific use cases

Needed a more flexible yet robust way to scale Python computational libraries

Capital One

Challenges

OSS Contribution Process

C4ML

Enterprise Contribution Process	Dask & RAPIDS Contribution Process

Enterprise Contribution Process

- Developer training on enterprise best practices for making contributions
- PRs reviewed internally by the enterprise leadership, Legal and Cyber Security
- Approved PRs published on public GitHub
- Iterations reviewed internally
- Trusted contributor status given at the repository level after a few PRs merged

Dask & RAPIDS Contribution Process

- Developer training on enterprise best practices and C4ML governance policy
- PRs reviewed by C4ML following the governance policy developed with OS team
- Approved PRs published on public GitHub
- Iterations reviewed internally
- Trusted contributor status given at the organization level
- Legal and the OS team audit periodically

Cloud Deployment Challenges

We operate in a restricted cloud environment.

- Restricted AWS Environment
  - We can only use whitelisted services
- No access to publicly hosted package repositories
  - Our internal package repositories only mirror common repositories
  - Can take upwards of 6 months for repositories to get mirrored
  - We must find other ways of installing key software
  - This is getting better, we are trying to improve the process

Dask Use Case - Deep Dive

Machine Learning Pipeline

C4ML

sample	to_csv	read_csv	feature engineering	to_csv	read_csv	merge	to_csv	read_csv	train	to_csv	read_csv	score

Airflow-orchestrated model training pipeline with a downsampled data set on a very large compute instance

```
>>> import pandas as pd

>>> df = pd.read_csv("training.csv")

>>> df = preprocess_data(df)

>>> model = xgb.train(df, ...)
```

XGBoost Model Stats

***40GB** training data set*

***~2.5 hour** training time per ensemble*

***~2.5 weeks** pipeline training*

Capital One

Initial Scaling with Dask

sample	to_csv	read_csv	feature engineering	to_csv	read_csv	merge	to_csv	read_csv	train	to_csv	read_csv	score

Utilized Dask dataframe to parallelize the sampling portion of the pipeline

Original Pipeline Stats

- ***Dataset merging is a compute-intensive problem and primed for distributed computing***
- ***300+ serialized joins in the data generation script***
- ***7 days to process ~1 TB dataset***

Pipeline Stats w/ Dask

***80** Dask workers*

***680 GB** of distributed memory*

***15 hour** processing time*

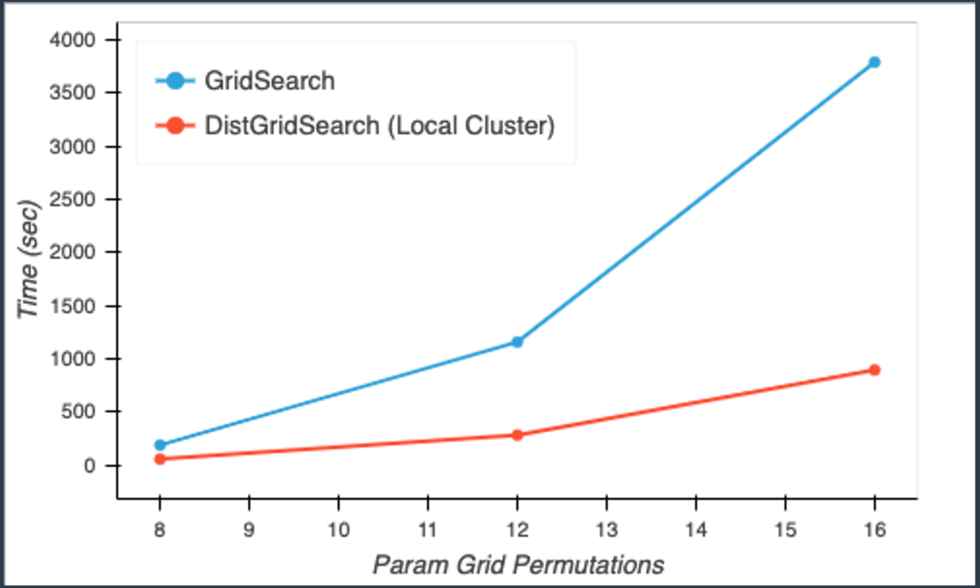
***91%** decrease in run time*

Further Scaling with Dask

C4ML

sample	to_csv	read_csv	feature engineering	to_csv	read_csv	merge	to_csv	read_csv	train	to_csv	read_csv	score

More Dask dataframes and dask-ml to parallelize feature selection and model tuning on highly utilized portions of the pipeline.



Pipeline Stats

- *Code used by many teams to build models*
- *Created shared Dask infrastructure*
- *Horizontal scaling and infrastructure agnostic*
- *Improved performance by parallel parameter searches*
- *Training on larger than memory datasets*

Capital One

Scaling with RAPIDS

C4ML

sample	to_csv	read_csv	feature engineering	to_csv	read_csv	merge	to_csv	read_csv	train RAPIDS	to_csv	read_csv	score

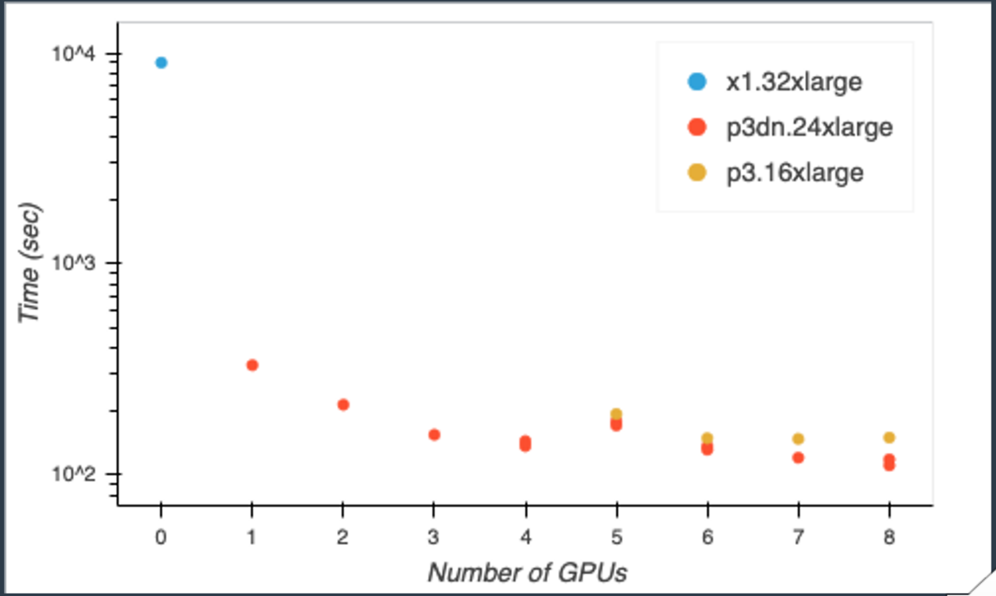
Use Dask and RAPIDS to scale XGB training on single-node, multi-GPU clusters

Pipeline Stats

***40GB** training data set / 4% of total*

***100x** speed up of training time for this dataset and model*

***~97%** reduction in training cost for this dataset and model*



CapitalOne

Scaling with RAPIDS

sample	to_csv	read_csv	feature engineering	to_csv	read_csv	merge	to_csv	read_csv	train RAPIDS	to_csv	read_csv	score

Use XGB training code as a real-world benchmark to test on multi-node, multi-GPU clusters

dataset	# nodes	node type	# GPUs	training size	test size	time	training cost
40 GB	1	x1.32xlarge	0	13 GB	6 GB	2 - 3 hrs	$26.68 - $40.01

Initial benchmark

Scaling with RAPIDS

C4ML

sample	to_csv	read_csv	feature engineering	to_csv	read_csv	merge	to_csv	read_csv	train RAPIDS	to_csv	read_csv	score

Use XGB training code as a real-world benchmark to test on multi-node, multi-GPU clusters

dataset	# nodes	node type	# GPUs	training size	test size	time	training cost
40 GB	1	x1.32xlarge	0	13 GB	6 GB	2 - 3 hrs	$26.68 - $40.01
40 GB	1	p3dn.24xlarge	8	13 GB	6 GB	2m 1s	$1.02
40 GB	1	p3.16xlarge	8	13 GB	6 GB	2m 23s	$0.82

Capital One

Scaling with RAPIDS

C4ML

sample	to_csv	read_csv	feature engineering	to_csv	read_csv	merge	to_csv	read_csv	train RAPIDS	to_csv	read_csv	score

Use XGB training code as a real-world benchmark to test on multi-node, multi-GPU clusters

dataset	# nodes	node type	# GPUs	training size	test size	time	training cost
40 GB	1	x1.32xlarge	0	13 GB	6 GB	2 - 3 hrs	$26.68 - $40.01
40 GB	1	p3dn.24xlarge	8	13 GB	6 GB	2m 1s	$1.02
40 GB	1	p3.16xlarge	8	13 GB	6 GB	2m 23s	$0.82
1.15 TB	12**	p3dn.24xlarge	96	288 GB	127 GB	52m 21s	$326.85

Scaling with RAPIDS

sample	to_csv	read_csv	feature engineering	to_csv	read_csv	merge	to_csv	read_csv	train RAPIDS	to_csv	read_csv	score

Use XGB training code as a real-world benchmark to test on multi-node, multi-GPU clusters

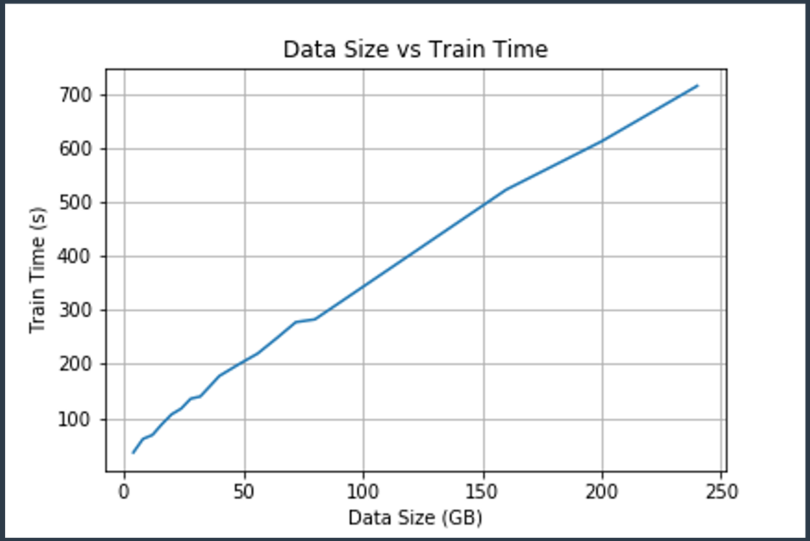
dataset	# nodes	node type	# GPUs	training size	test size	time	training cost
40 GB	1	x1.32xlarge	0	13 GB	6 GB	2 - 3 hrs	$26.68 - $40.01
40 GB	1	p3dn.24xlarge	8	13 GB	6 GB	2m 1s	$1.02
40 GB	1	p3.16xlarge	8	13 GB	6 GB	2m 23s	$0.82
1.15 TB	12**	p3dn.24xlarge	96	288 GB	127 GB	52m 21s	$326.85
1.15 TB	2**	p3dn.24xlarge	16	288 GB	127 GB	22m 47s	$54.47

Experiments with RAPIDS

C4ML

sample	to_csv	read_csv	feature engineering	to_csv	read_csv	merge	to_csv	read_csv	train RAPIDS	to_csv	read_csv	score

Exploring the performance of scaling the amount of data on a single p3dn.24xlarge Instance



A graph demonstrating how train time is affected by data size on a single instance with multiple GPUs

Data Scaling

- **500,000 to 30,000,000 rows and 493 feature columns (4GB to 240GB)**
- **Train time scales linearly with data size on a single instance as expected**

Capital One

Experiments with RAPIDS

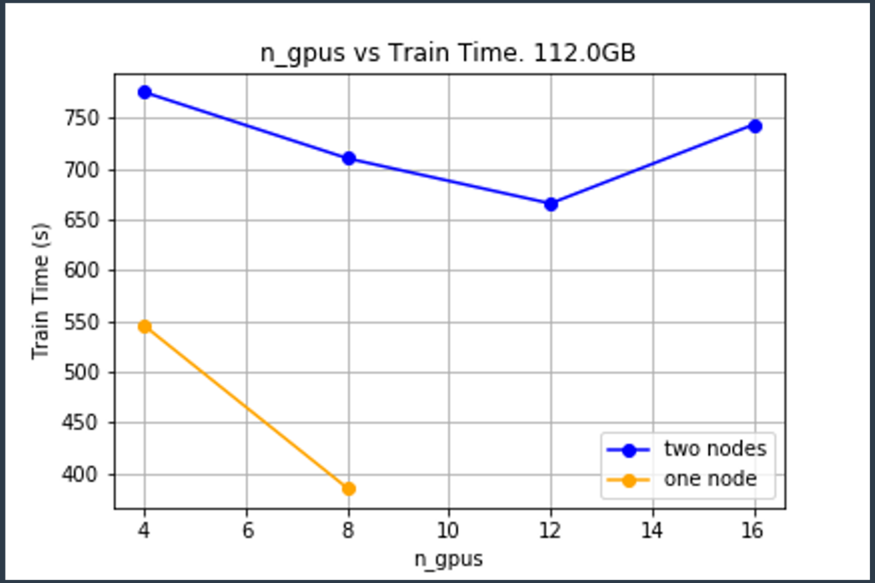
C4ML

sample	to_csv	read_csv	feature engineering	to_csv	read_csv	merge	to_csv	read_csv	train RAPIDS	to_csv	read_csv	score

Exploring the performance of scaling the number of GPUs with regards to a static data size

GPU Scaling

- **112GB of data, close to 90% of the maximum amount of data that 4 32GB GPUs can hold**
- **Two tests:**
  - **Single Instance, test 4 and 8 GPUs**
  - **Multi Instance, test from 4 to 16 GPUs at an interval of 4 GPUs**
- **~42% increase in training time with 4 GPUs on a single instance vs 4 GPUs split across two Instances**
- **~84% increase in training time with 8 GPUs on a single instance vs 8 GPUs split across two Instances**

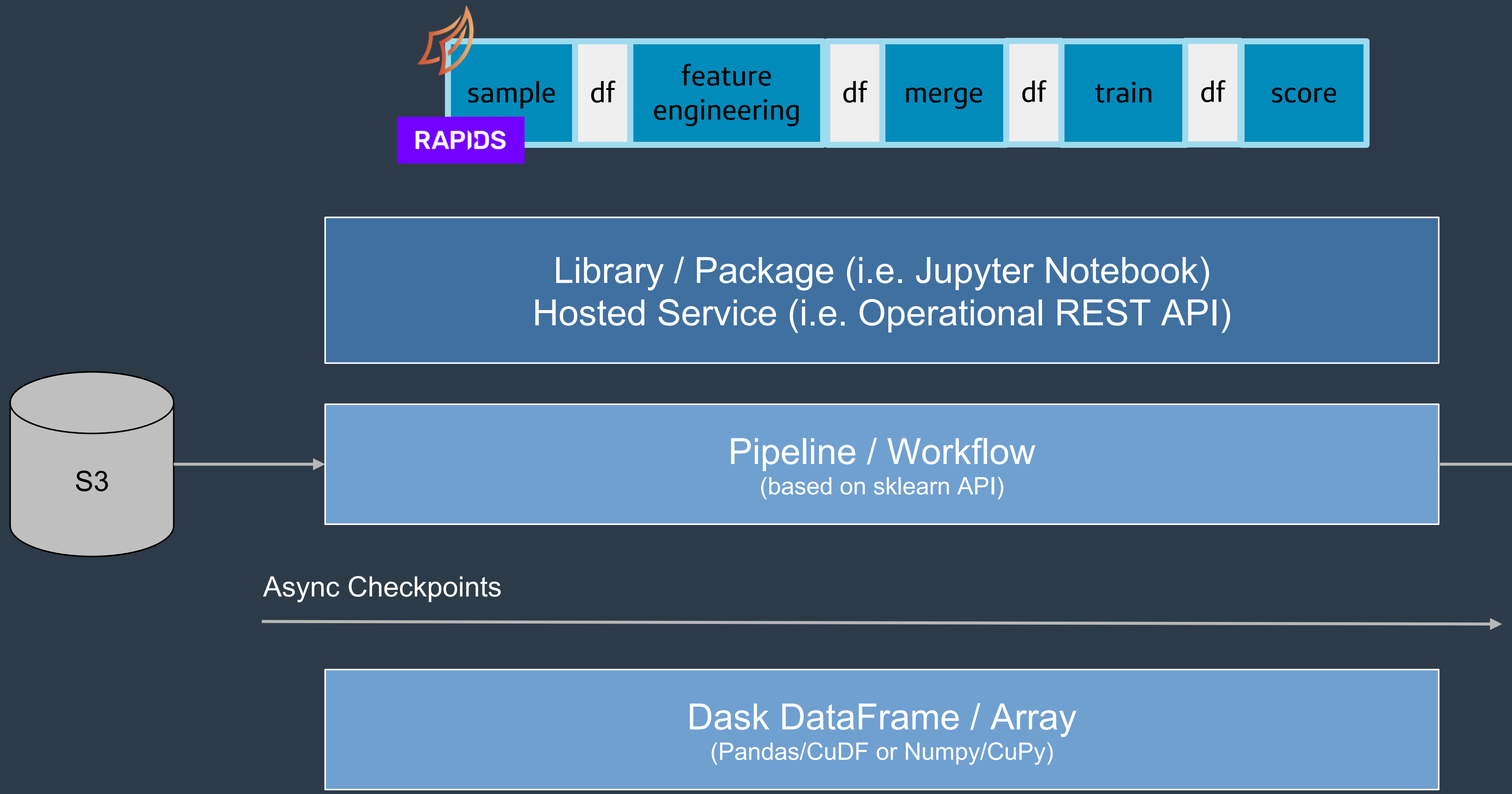


A graph comparing the effect of the number of GPUs on training time using 112 GB of data.

Capital One

Optimal State

Optimal Pipelines



Adopting the sklearn API

C4ML

Single Core

```
>>> import pandas as pd
>>> df = pd.read_csv("training.csv")
>>> X = df.drop(target, axis=1)
>>> y = df[[target]]

>>> from package.model_selection import CustomCV
>>> import xgboost as xgb
>>> clf = xgb.XGBClassifier()

>>> ccv = CustomCV(clf)
>>> ccv.fit(X, y,
...  early_stopping_rounds=4,
...  eval_metric=["auc", "logloss"]
...  )
```

Dask

```
>>> import dask.dataframe as dd
>>> df = dd.read_csv("training.csv")
>>> X = df.drop(target, axis=1)
>>> y = df[[target]]

>>> from package.model_selection import CustomCV
>>> import xgboost.dask as dxgb
>>> clf = dxgb.XGBClassifier()

>>> ccv = CustomCV(clf)
>>> ccv.fit(X, y,
...  early_stopping_rounds=4,
...  eval_metric=["auc", "logloss"]
...  )
```

RAPIDS

```
>>> import dask_cudf as cdd
>>> df = cdd.read_csv("training.csv")
>>> X = df.drop(target, axis=1)
>>> y = df[[target]]

>>> from package.model_selection import CustomCV
>>> import xgboost as xgb
>>> params = {'n_gpus': 1}
>>> clf = xgb.XGBClassifier(**params)

>>> ccv = CustomCV(clf)
>>> ccv.fit(X, y,
...  early_stopping_rounds=4,
...  eval_metric=["auc", "logloss"]
...  )
```

Scale by changing a few lines of code

Capital One