CLOUDERA

WHITE PAPER

INTRODUCING AN END-TO-END BLUEPRINT FOR CHURN PREDICTION AND MODELING

SPEED UP YOUR MACHINE LEARNING WORKFLOWS WITH GPU ACCELERATION ON THE ENTERPRISE DATA CLOUD

By William Benton, Principal Product Architect, NVIDIA and
Jacob Bengtson, Sr. Technical Product Marketing Manager, Cloudera

NVIDIA

Table of Contents

Introduction

If you want to learn how to build a predictive model to solve a particular kind of business problem, you'll likely have no trouble finding a tutorial showing you how to extract features and train a model. This is a pleasant side effect of the popularity and importance of data science and machine learning across industries. However, solving business problems with machine learning isn't just about training models or even about finding the best features; if you want a starting point for solving an entire problem from end to end, or if you want a realistic production workflow to test your system architecture or hardware performance, these tutorials leave many of the hard parts as an exercise for the reader.

In this document, we're going to introduce a complete solution for predicting customer churn — that is, answering the question, "Given what we know about this customer, is she likely to not renew her contract?" We'll also show the benefits of running this workflow on the Cloudera Data Platform and accelerating each stage of this workflow with NVIDIA GPUs. If you're interested in learning more, you'll be able to register for a full eBook that will provide a deep dive on how to make the most out of this amazing combination, including analytic processing and federation of structured data, integrating enterprise data engineering pipelines with machine learning, and accelerated inference.

We're going to start by level-setting: introducing a typical end-to-end machine learning workflow, describing some of the challenges of production machine learning systems, and explaining some of the specific concerns in understanding customer churn. We'll then introduce the overall architecture for our churn prediction solution.

Data Science Workflows and the Customer Churn Problem

If we were talking about data science ten years ago, we'd have been referring to a broad discipline that combined domain expertise with elements of analytics, applied statistics, machine learning, computer science, and software engineering. A data scientist might have had to wear many hats: identifying business objectives; cleaning data; processing big data at scale; identifying features; encoding features; selecting and training models; building applications, reports, or dashboards that incorporate those models; and even managing infrastructure! Today, a typical data scientist is more specialized and only focuses on parts of the classic data science workflow: characterizing data, finding patterns, and training models. The other parts of the classic data science workflow are still important, but the practitioners who are responsible for them may have a range of titles other than "data scientist," such as data engineer, application developer, machine learning engineer, or MLOps engineer.

This shift of responsibilities comes as data platforms like Cloudera Data Platform (CDP) evolve to offer new capabilities and integrations. CDP provides tools to address the entire machine learning lifecycle, with seamless connections to data sources across your infrastructure.

These changes create opportunities for new workflows. We'll introduce such a workflow now (borrowing concrete terminology from this paper) in the context of the churn prediction problem.

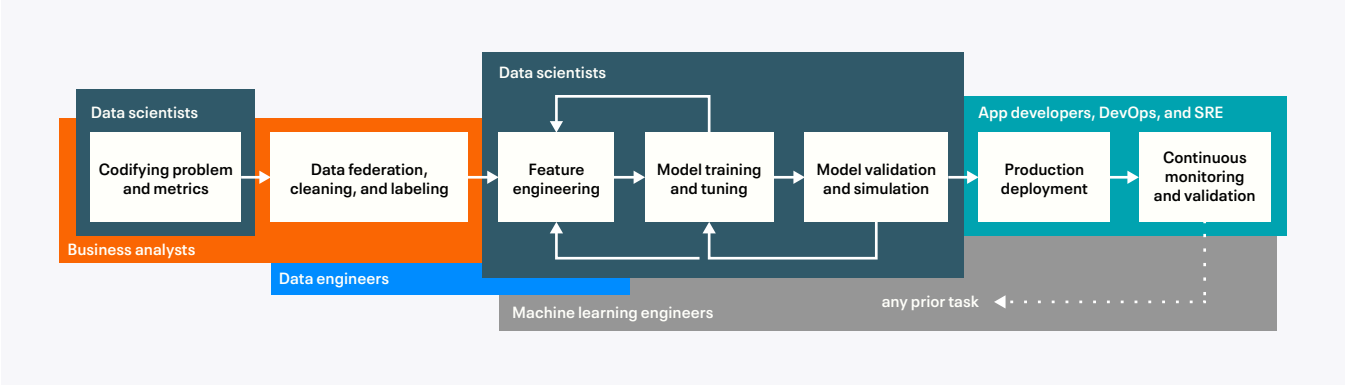

Figure 1. A typical data science workflow, showing the human processes from discovery through production, and the team members and roles involved in each stage.

Codifying our Problem and Defining Metrics

The first step is formalizing the problem we're trying to solve and defining metrics of success. On a long enough timeline, every customer will fail to renew their subscription, but producing a model that asserts "yes, eventually" for any customer isn't useful. Similarly, we might be able to predict quite accurately that a customer who has begun the process to cancel her account is quite likely to churn in the near future, but such a prediction is also trivial. Ideally, we'd want a model that identifies signals in a customer's profile that indicate that the customer may churn in the near but not immediate future and is thus a good candidate for targeted retention efforts. (We may also want to rank customers for more costly targeted retention efforts, e.g., by expected future lifetime account value.) The exact details of how we formally define churn will be specific to our business and the kinds of retention efforts we may have to offer; given such a definition, though, we can label historical customer records and build a target for our model.

Analyze structured and unstructured data at scale

- CDP Data Warehouse offers a cloud-native data warehouse fully integrated with streaming, data engineering, and machine learning analytics.
- Cloudera Data Lake Service provides capabilities for governing, managing, securing and auditing data in a data lake.

Orchestrate and automate complex data pipelines

- CDP Data Engineering is the only cloud-native service purpose-built for enterprise data engineering teams. Building on Apache Spark, Data Engineering is an all-inclusive data engineering toolset that enables orchestration automation with Apache Airflow, advanced pipeline monitoring, visual troubleshooting, and comprehensive management tools to streamline ETL processes across enterprise analytics teams.

Accelerate the process of putting machine learning to work

- CDP Machine Learning is a scalable, open ML platform that helps streamline the process of getting analytic workloads into production and intelligently manage machine learning use cases across the business at scale.

Federating data in a single logical location, like a data lake, is an important step for further processing.

Feature engineering: The process of building a technique to transform structured data into data that we can pass to a model training algorithm or to a machine learning model.

Data Federation, Cleaning, Analytics, and Labeling

Our focus next shifts to identifying, transforming, and federating potentially-relevant information about our customers. Ideally, we will be able to draw upon structured data, such as transactional databases, and unstructured data, such as call center transcripts. Federating this data in a single logical location, like a data lake, is an important step for further processing. We then need a way to ensure that the data we have meets a given standard for quality — for example, are all of the values in their expected ranges? Are certain records missing important values? — and impose these constraints while processing raw data from the data lake and storing structured data in a data warehouse. Given a source of clean, structured data, we can support exploratory analytics, report generation, and ultimately machine learning model training. In many cases, ad hoc queries and reports are related to the problems we might want to solve with machine learning: for example, quarterly reports will likely cover net loss or gain in subscribers and revenue, and a database programmer or business analyst might use interactive queries to identify attributes of customer records that are correlated with various business outcomes.

Data in our data warehouse is cleaned, federated, structured, organized, and (can be) labeled with various outcomes of interest for each customer — these are all properties that make it easier to find patterns and business value in the data. But relational databases are also typically organized with normal forms, which means that all of the relevant data about a given customer may be spread across multiple more-or-less independent tables. In order to prepare to train a machine learning model, we'll need to denormalize our data, and go from long-form tables of individual observations of a certain kind to wide-form tables whose rows include all of the relevant data we have for each customer.

Feature Engineering

The wide-form data we have now is analogous to rows in a database table or programming-language objects — it's structured in a convenient format for further programmatic manipulation, but it's not in the most convenient format for training a machine learning model. The process of feature engineering is the process of building a technique to transform structured data into data that we can pass to a model training algorithm (or to a machine learning model). It entails identifying techniques to map from structured data to points in multidimensional space in such a way that the mapping preserves some interesting and meaningful structure of the source data; concretely, we might choose features, or attributes of each customer, and encode them as numbers so that we can encode customers as vectors of numbers in such a way that similar customers map to similar vectors and (ideally) there is a relatively straightforward way to partition the feature space between vectors for customers that churned and those for customers that have not yet churned. Once we've identified which features are important and how to encode them, we can develop a feature extraction pipeline that we can use to prepare structured data for model training or inference.

Model Selection, Training, Tuning, and Validation

At their core, machine learning models provide compact, useful summaries of datasets. In the case of the customer churn problem, our model will optimize a function to identify churning customers by identifying combinations of features that imply most strongly that a customer will churn. In order to train a model, we'll need to identify a modeling approach, tune the parameters (called hyperparameters) that govern its behavior, and evaluate its performance before ultimately validating that it has generalized by testing its performance on held-out (and thus novel) data.

Follow best practices for ML management and governance

- CDP Machine Learning's **MLOps capability** enables one-click model deployment, model cataloging, and granular prediction monitoring to keep models secure and accurate across production environments.

CONCEPT DRIFT

When novel data diverge enough from training data that model performance begins to materially suffer.

Production Deployment, Monitoring, and Feedback

Even a promising model can only be as useful as the system that employs it to solve a business problem. In order to deploy our model into production, we'll need:

1. a way to reproduce the feature extraction pipeline and model training pipelines from our discovery workflow in a production environment as a production training pipeline (that is, a way to take raw, labeled data and produce a trained model), and
2. a way to publish our feature extraction pipeline and the trained model itself as a production inference pipeline (that is, a service that takes a record of novel data, extracts features from that record, and then uses the model to make a prediction about it)

In addition, we'll need to monitor our model's inputs and performance to account for concept drift, which occurs when the novel data we see have diverged enough from the data we trained our model on that its performance begins to materially suffer. (As a concrete example, a mobile network operator might start losing customers who spend a lot of time internationally roaming after a competitor introduces a free or reduced-cost international roaming plan – the underlying cause and phenomenon wouldn't have been captured in any training data collected before the competitive landscape changed.) Other changes that could impact the performance of our model include changes to upstream data formats or schemas, the introduction of new customer categories, and long-term trends in the overall market.

Identifying concept drift is one of many problems that can cause us to return to an earlier stage and revisit engineering and modeling decisions that we made. At the end of this workflow, we may have a model with excellent predictive performance that ultimately doesn't enable us to satisfy the right business metrics, requiring us to reevaluate how we formalized our problem. Concept drift may require retraining a model with additional training data. Unanticipated problems in production may suggest using a different overall approach. Finally, the processes of feature engineering and model training are often iterative since decisions made about how to encode features impact the kinds of models that we can effectively train.

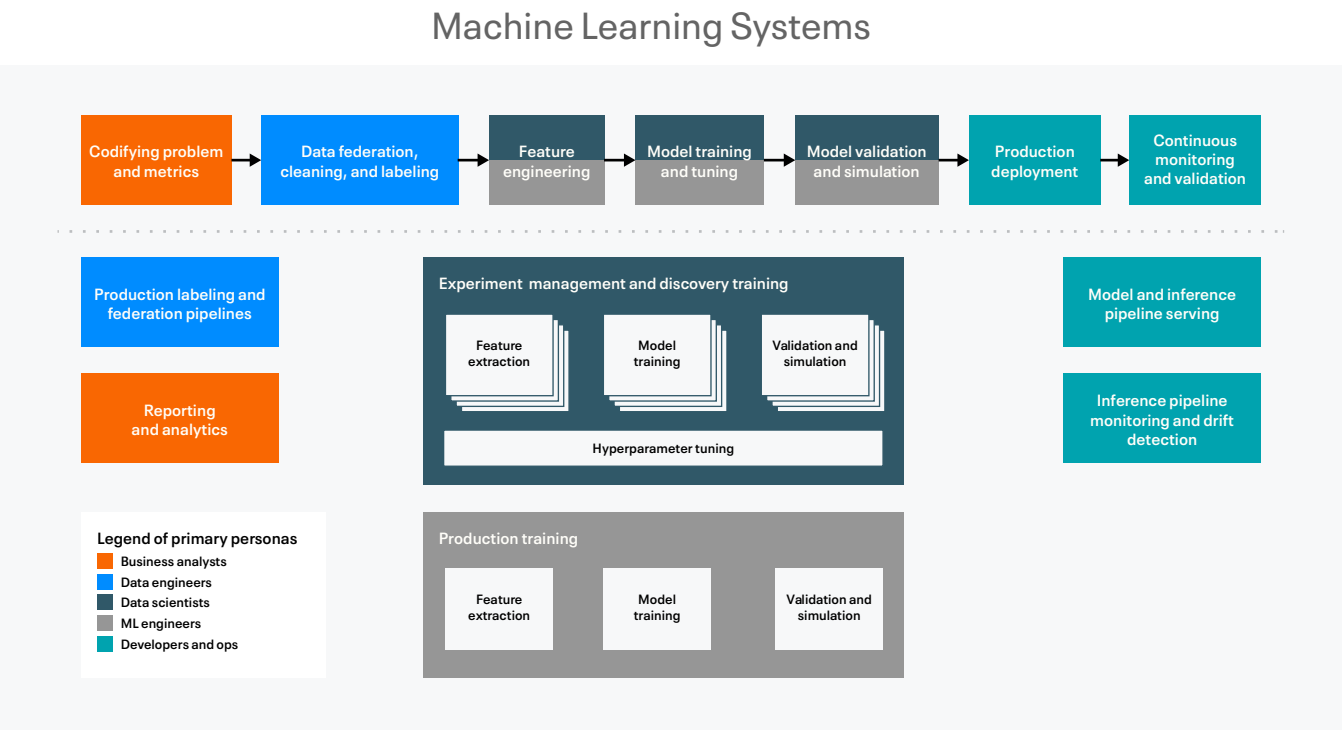

Figure 2. A mapping from machine learning workflow stages to the machine learning system components each informs.

Data feedback is an important part of machine learning. New data collected during operation can feed back to the data federation pipeline for future training.

We've just discussed a human process by which teams develop machine learning systems, but we haven't explicitly described the machine learning systems they'll create. Each of the stages we discussed in the discovery workflow informs some parts of a machine learning system. There's a high-level view of one such mapping in the figure above, color-coded by the personas involved in each stage.

Some correspondences between human tasks and system components are obvious. For example, the feature engineering approach a data scientist chooses will directly inform the code that executes as part of a production feature extraction pipeline. The modeling approaches that are most successful in prototypes and experiments will inform the modeling approaches used in production, and so on.

Some correspondences are more interesting. Just as the discovery workflow is iterative and lessons learned in later stages can feed back to earlier stages, explicit data feedback is an important part of machine learning systems: new (potentially-labeled) data collected during the operation of a system can feed back to the data federation pipeline to be used for future training efforts, metrics about model performance and business metrics can be tracked in a single dashboard (as well as monitored for evidence of concept drift), and experimental approaches can be evaluated in production in parallel to established ones.

Overall Application Architecture

We'll be building up a complete system for churn modeling and prediction, consisting of four applications that work together: a data federation application that integrates structured data from a data warehouse, an analytics application that prepares human- and machine-readable reports from the federated data in order to generate business insight and support a data scientist's workflow, a feature extraction and model training application that takes flat training data and reports and generates a trained model and an inference pipeline, and an inference service that takes information about a customer and predicts whether or not that customer will churn.

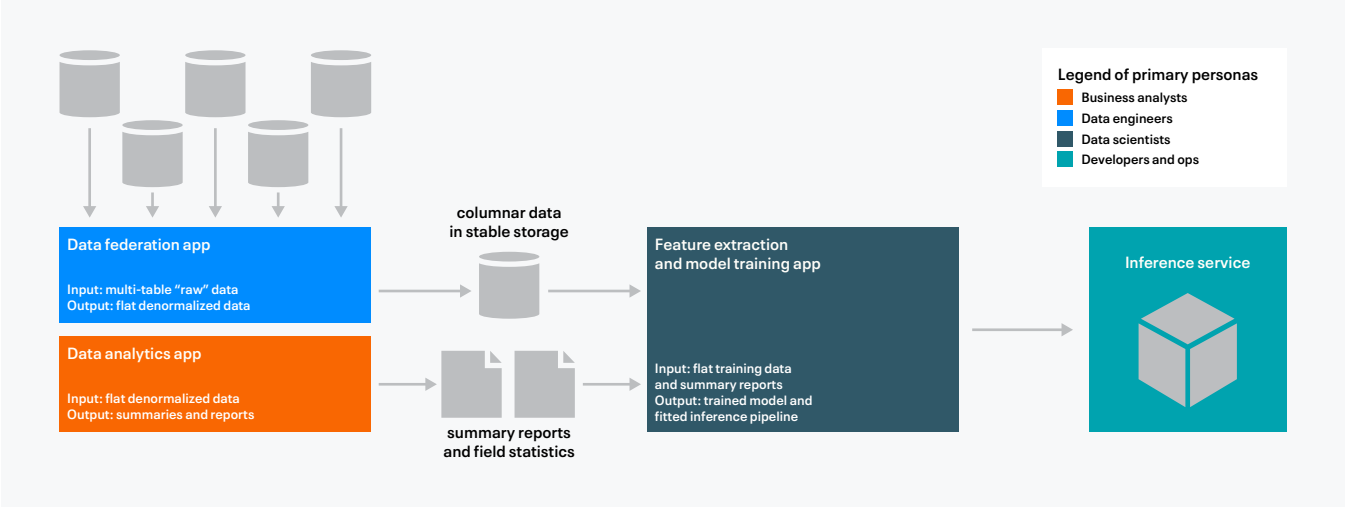

Figure 3. A mapping from machine learning workflow stages to the machine learning system components each informs.

CLOUDERA DATA PLATFORM AND NVIDIA

To save time with model training, data scientists often turn to NVIDIA GPUs to accelerate machine learning and deep learning workloads. With Cloudera Data Platform (CDP), practitioners can leverage best-in-class GPU computing frameworks from NVIDIA natively on any cloud with CDP Public Cloud and/or on-premises with CDP Private Cloud.

As we've mentioned, we're going to focus on some of the parts of machine learning systems that are often ignored in machine learning technique tutorials: accelerating data federation, query processing, and exploratory analytics; managing the connections between different system components developed by different teams in different languages; and making the best use of our compute resources across the lifecycle of our system.

Technology Stack

For our data federation and analytics applications, we'll be using Apache Spark and the RAPIDS Accelerator for Apache Spark, which enables us to accelerate Spark data frame operations on NVIDIA GPUs. Our feature extraction and model training application will use accelerated libraries from RAPIDS and the Python data ecosystem, including cuDF, cuML, Dask, and XGBoost. We'll use the RAPIDS Forest Inference Library to accelerate inference. There are some important differences in how these libraries work and how they achieve GPU acceleration; we'll briefly examine each.

RAPIDS

The RAPIDS libraries provide GPU-accelerated implementations of familiar interfaces from the Python data ecosystem. RAPIDS users are thus able to benefit from GPU acceleration without relying on implementing custom compute kernels or manually managing parallelism or transfers between host and device memory. cuDF is a GPU-accelerated data frame library, which allows users to manipulate collections of typed, structured data with an interface similar to the popular pandas library. cuML is a GPU-accelerated machine learning library that includes building blocks for feature extraction and model training pipelines that implement the scikit-learn estimator interface. cuML also includes a SHAP implementation for model explainability and FIL, an accelerated library for production inference with tree ensemble models. In addition, these libraries can share on-GPU data with other machine learning and deep learning libraries through the CUDA array interface. RAPIDS includes other libraries as well; the following figure shows how these map to the personas involved in the data science discovery lifecycle.

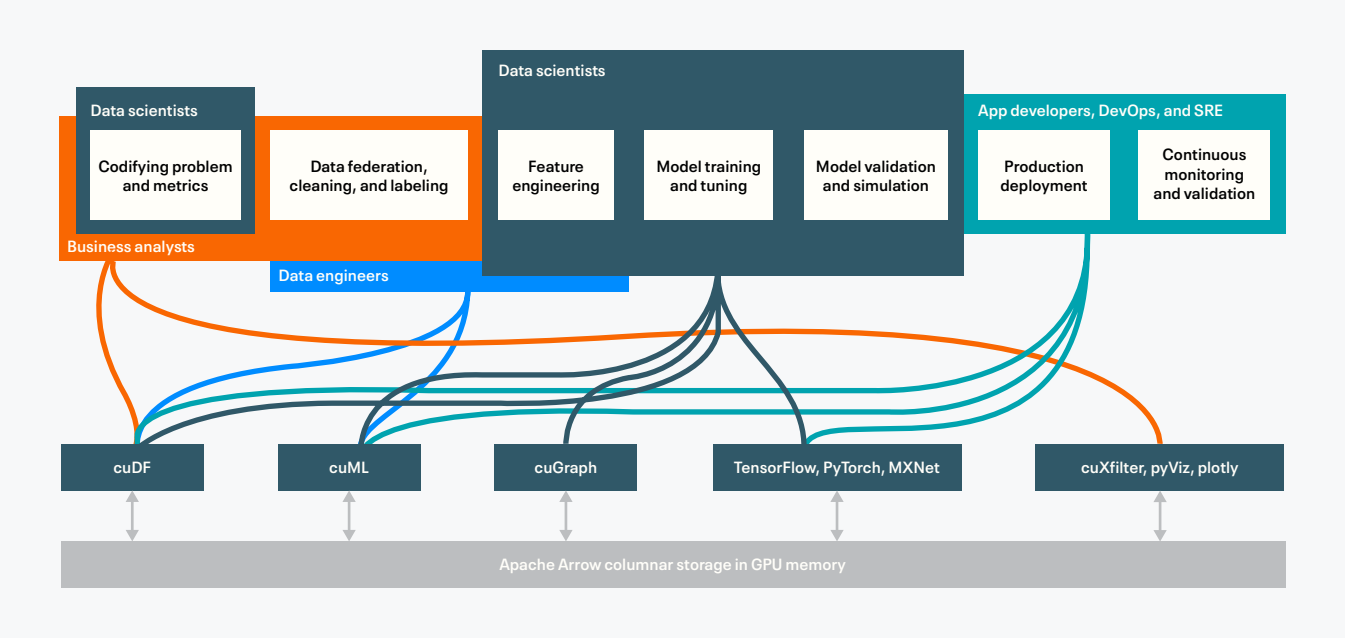

Figure 4. A mapping from personas and workflow stages to RAPIDS libraries, including cuDF, cuML, and cuGraph, and ecosystem projects that interoperate with these.

While cuDF and cuML provide familiar interfaces for data scientists, they are not drop-in replacements for pandas and scikit-learn respectively. Not every operation or algorithm in pandas and scikit-learn is amenable to GPU acceleration (or parallel execution in general); the RAPIDS libraries focus on implementing operations that can be accelerated on GPUs. A true drop-in replacement would need to transparently fall back to serial implementations for operations that won't parallelize; since it's possible, for example, to include serial scikit-learn estimators and transformers in a machine learning pipeline that uses cuML for performance-sensitive stages, it's possible for users to explicitly use serial implementations when necessary. In addition, most of the algorithms in RAPIDS are designed for a single node and a single GPU; in order to scale out or use more memory than is available in a single GPU, users will need to use the scale-out framework Dask in conjunction with cuDF and cuML, as in the following figure.

PUSH ML PERFORMANCE BOUNDARIES

The RAPIDS Accelerator for Apache Spark is now integrated with CDP Private Cloud Base, enabling enterprises to accelerate data pipelines and push the performance boundaries of data and machine learning (ML) workflows. Drive faster AI adoption and deliver better business outcomes without changing any code.

The RAPIDS Accelerator for Apache Spark

The RAPIDS Accelerator for Apache Spark takes a different approach to accelerating data science workloads on GPUs. Fundamentally, its approach is to provide transparent acceleration of Spark data frame jobs via a Spark plugin that integrates with Spark's query planner. The plugin rewrites data frame query plans in order to evaluate accelerable operations with implementations that use libcudf (the C++ library providing accelerated data frame functionality to the Python cuDF library) to execute on the GPU. Operations that cannot be accelerated will run on the CPU with Spark's built-in implementations; if there is a branch of a query plan that includes both accelerable and non-accelerable operations, the RAPIDS Accelerator plugin will automatically insert transfers between host and device memory so that both kinds of operations can work together transparently to execute a given query plan. The RAPIDS Accelerator for Apache Spark also provides an accelerated shuffle implementation using UCX (for data transfer within clusters) and integration with GPU-accelerated XGBoost.

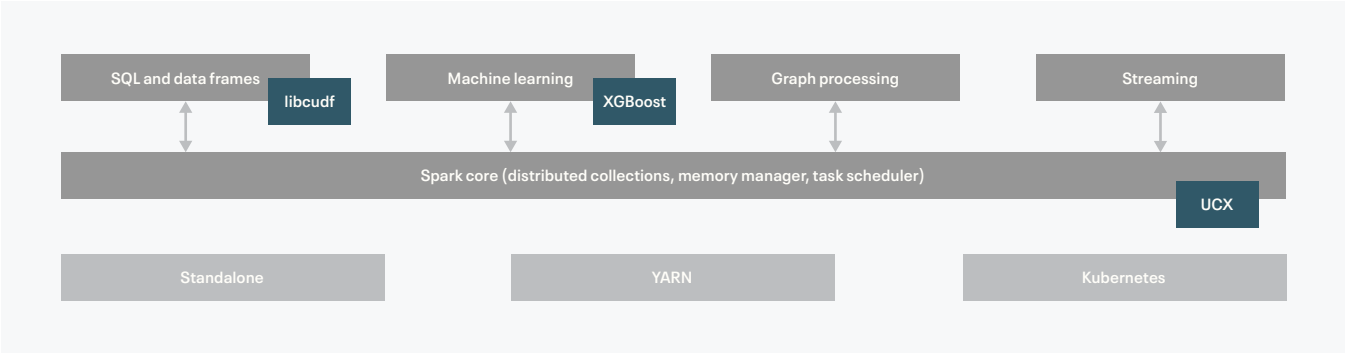

Figure 5. The architecture of GPU-acclerated Spark. Gray blocks in the bottom layer are the underlying resource managers Spark works with; the middle block is Spark's core task scheduler, resource manager, and low-level distributed collection API; the blocks on top are Spark's high-level APIs. Purple boxes indicate where extensions can provide GPU acceleration to Spark applications.

Transparent acceleration has both a benefit and a cost for users. The benefit of transparent acceleration is that users can expect that a Spark application that runs successfully on the CPU will also run successfully with the RAPIDS Accelerator for Apache Spark enabled. The cost of transparent acceleration is that the performance improvement any given application can expect may be difficult to predict and will be a function of several factors: what percentage of its runtime it spends in accelerable data frame operations, how much work can be performed on the GPU between CPU-only operations, and how much each accelerable operation can improve when running on the GPU. Paradoxically, in order to take full advantage of transparent acceleration, data engineers and application developers may need to consider which parts of their application can be accelerated in order to make small changes for maximum performance. The following figure shows part of a query plan in which some operations are accelerated but one aggregate operation runs on the CPU.

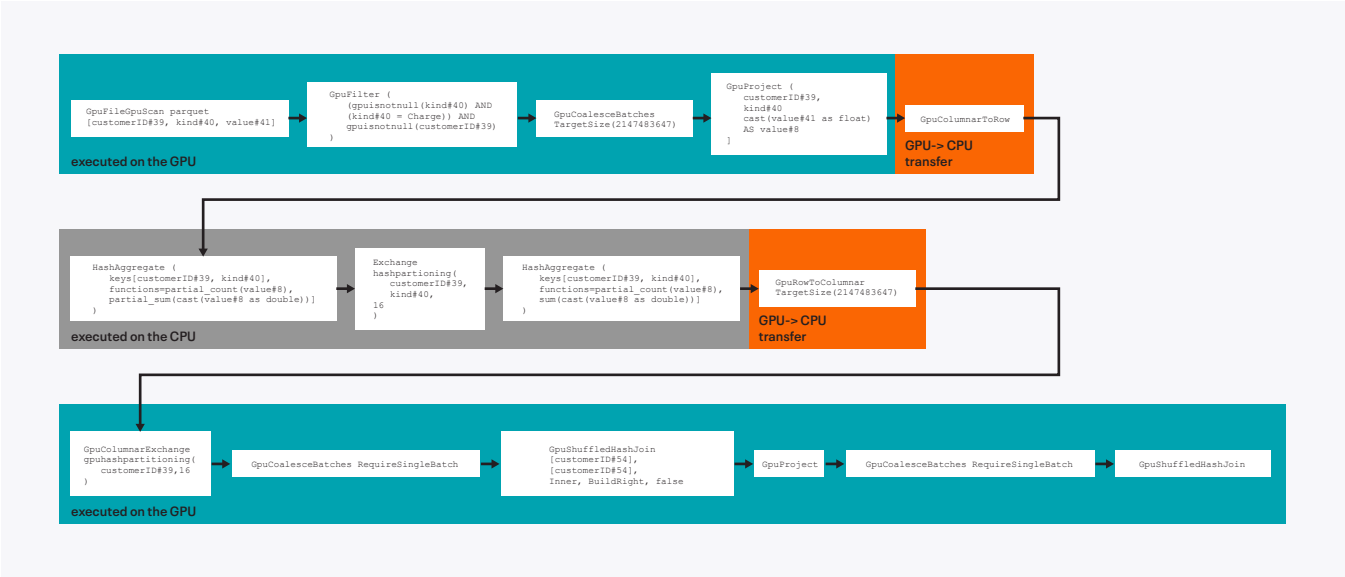

Figure 6. A subset of an accelerated Spark query plan, showing GPU-accelerated portions, host-device and device-host transfers, and operations falling back to execute on the CPU.

Finally, transparent acceleration depends on applications using Spark's data frame and query abstractions. Because data frame operations are specified in an expressive but high-level API, it is possible to reorganize and rewrite query plans before they are executed, including replacing operations with higher-performance implementations (Spark itself takes advantage of this by generating native code to execute portions of query plans). Indeed, all data frame operations and SQL queries are planned and transformed by Spark before execution, and Spark itself provides a plugin layer so that external code (like the RAPIDS Accelerator) can alter the behavior of the query planner. Spark's lower-level resilient distributed dataset (RDD) API allows users to execute arbitrary code on partitioned distributed collections, but this additional flexibility of expression for application developers comes at the cost of flexibility of execution for Spark itself: since it is not, in general, feasible to safely transform arbitrary host-language code, Spark must treat the functions passed into its RDD API as opaque.

The Cloudera Data Platform

CDP is the industry's first enterprise data cloud. The platform manages and secures data workloads across all major public clouds and on premise in the private cloud. CDP also enables a seamless connection between the public and private cloud for a hybrid experience.

CDP ships with rich capabilities to power your entire data lifecycle, from the edge to AI, including distributed and governed storage, data engineering pipelines, and exploratory or production machine learning.

RAPIDS and the RAPIDS Accelerator for Apache Spark are now available for your enterprise data lake as part of the Cloudera Data Platform. In the full eBook, we'll show you how this ML application for predicting customer churn can be implemented on CDP Private Cloud Base and leverage these best-in-class GPU computing frameworks.

About Cloudera
At Cloudera, we believe that data can make what is impossible today, possible tomorrow. We empower people to transform complex data into clear and actionable insights. Cloudera delivers an enterprise data cloud for any data, anywhere, from the Edge to AI. Powered by the relentless innovation of the open source community, Cloudera advances digital transformation for the world's largest enterprises.

Learn more at cloudera.com
US: +1 888 789 1488
Outside the US: +1 650 362 0488

Connect with Cloudera
About Cloudera:
cloudera.com/more/about.html

Read our VISION blog:
vision.cloudera.com

Follow us on Twitter:
twitter.com/cloudera

Visit us on Facebook:
facebook.com/cloudera

See us on YouTube:
youtube.com/user/clouderahadoop

Join the Cloudera Community:
community.cloudera.com

Read about our customers' successes:
cloudera.com/more/customers.html

Make an Impact on Your Business with ML That Scales

We've discussed our tech stack: RAPIDS, the RAPIDS Accelerator for Apache Spark, and NVIDIA GPUs. With the Cloudera Data Platform (CDP), you can take advantage of these accelerated libraries, but on the industry's first enterprise data cloud.

For our application, using commodity servers with NVIDIA GPUs made data federation nearly 2x faster, improved analytics performance by nearly 7x, and made machine learning model training faster on a cluster or possible on a single node (yet still dramatically faster) – resulting in over a 3x improvement in performance-per-dollar relative to CPU-only servers on our end-to-end workload. Speed and convenience aren't enough for enterprises, though – your business needs trusted partners and robust support.

CDP delivers an enterprise data cloud with open, hybrid data architecture, seamlessly connecting your data across your fragmented IT landscape. It also features an always-on Shared Data Experience (SDX) layer that enables holistic security, governance, and compliance across the full data lifecycle.

Cloudera Machine Learning (CML) for CDP enables enterprise data science teams to collaborate across the ML lifecycle with immediate access to enterprise data pipelines, scalable compute resources, and access to preferred tools.

With SDX integrated into CML, you can deploy models to production with access, governance, and security rules inherited directly from CDP.

CML also features Applied Machine Learning Prototypes (AMPs), a catalog of end-to-end reference ML projects that you can deploy at the click of a button. AMPs help your company realize ROI from ML faster and at a greater scale.

Start your free test drive at Cloudera.com.

Cloudera, Inc. 5470 Great America Pkwy, Santa Clara, CA 95054 USA cloudera.com