2024년 10월 17일

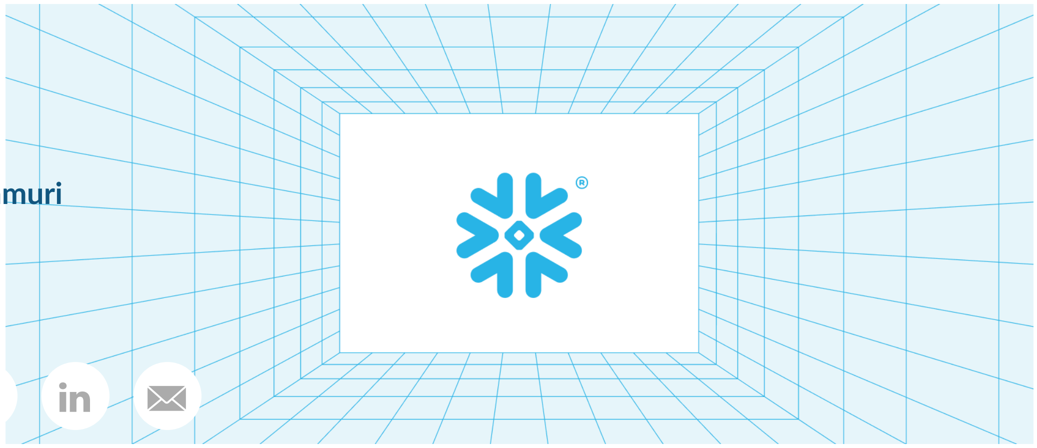
AUTHOR

Allie Fero

Sravankumar Nandamuri

How We Built It: Optimizing Snowflake ML Container Runtime for Fast, Scalable Data Loading and Model Training

Machine Learning



SHARE

Snowflake recently launched the public preview of Container Runtime, a fully-managed container environment that distributes processing on CPUs and GPUs for the most popular Python libraries and frameworks. Training models with Snowflake Notebooks on Container Runtime delivers 3-7x faster performance compared to running the same workload with your Snowflake data using open source libraries outside of the runtime.

This fully-managed, container-based runtime meets the most demanding ML needs of Snowflake customers. It is built for faster, more scalable data loading and training, with optimized data ingestion that scales quickly and efficiently for large data sets and support for multiple distributed trainers. In this blog, we take a deep dive into how we built an easy-to-use, high-performing container environment that helps data scientists and ML engineers focus on model development and impact, not the drudgery of infrastructure, interoperability, and scaling.

Container Runtime Architecture

Snowflake’s Container Runtime sits on top of Snowpark Container Services (SPCS), Snowflake’s fully-managed container service. It provides a prebuilt, containerized compute environment that works natively with Snowflake Notebooks.

When you start your Notebook, we create a new SPCS service that uses the Snowflake ML Container Runtime as the image. The service is provisioned using one node in the compute pool; if there are not enough active nodes, the compute pool will *automatically* bring up a new node to run the Notebook, up to the maximum nodes allowed. This automatic scaling of compute means that you can focus on solving your problem instead of managing the underlying infrastructure.

AUTHOR

Allie Fero

Sravankumar Nandamuri

Container Runtime features three main components:

- **Prebuilt Python environment** with up-to-date versions of the most popular ML/DS packages already installed
- **Managed, stateful distributed-compute-orchestration service** running on top of SPCS
- **Snowflake ML APIs** to facilitate model training scale-out and data ingestion

SHARE

Figure 1. Container Runtime architecture stack.

Easy and Performant: Container Runtime Features

Optimized data ingestion

Container Runtime is a flexible, generic computing environment, where users can run open source code and Snowflake-specific APIs. As such, we make it easy and performant to convert your Snowflake data into open source data objects, like pandas DataFrames or PyTorch Datasets. To convert data from Snowflake into open source formats, you can leverage Snowflake's Data Connector API.

Benefits of the DataConnector API include:

> **Easy to Use**: Converts Snowflake data directly to open source data objects
>
> **Unified**: DataConnector uses runtime-specific implementation in Container Runtime, and Arrow implementation outside, so you can use the same code inside and outside of Snowflake
>
> **Performant**: The runtime-specific implementation has been designed to maximize the throughput of Snowflake data into your Snowflake training workloads or directly into third-party data formats

Diving into performance, the Data Connector, with its specialized implementation for a runtime environment, is built to utilize all of the compute resources available to read and load the data in parallel. It is currently implemented using Ray Data to take advantage of multiprocessing and a shared object store, with additional optimizations to even out batch sizes and batch data in memory for optimal performance.

Figure 2. Data Connector system architecture diagram. The dependency inversion of the data ingester allows us to inject a runtime-specific implementation.

In Container Runtime, data ingestion takes the following steps:

1. Materialize data as a Snowflake Dataset or Snowflake Result Batch
2. Define a Ray Datasource that can read from Snowflake data
3. Leverage Ray Data to load into distributed memory store and convert to the appropriate data format for downstream workload

AUTHOR

Allie Fero

Sravankumar Nandamuri

SHARE

AUTHOR

Allie Fero

Sravankumar Nandamuri

SHARE

Figure 3. Steps to ingest data from Snowflake into a Ray Dataset.

By using Ray Data, we benefit from its use of asynchronous, distributed tasks that run in dedicated Python processes. This gives us the flexibility to leverage as many compute resources as necessary to load and transform large data sets efficiently. In addition, we layer on some additional optimizations to ensure high performance with Snowflake specific data sources:

When loading from a Snowpark DataFrame, we customize the data source to re-batch result batches to be equally sized for efficient parallelization across data ingestion processes.

When transforming data from PyArrow tables to NumPy arrays, we buffer batches in memory for wide data sets. This prevents inefficient Numpy concatenations across columns from being repeated on a per-batch basis.

The Data Connector API is used by default with all Snowflake trainers, but users can leverage it directly to transform their data into third-party data objects, such as a pandas DataFrame as well.

Average Throughput by Data Set Size

AUTHOR

Allie Fero

Sravankumar Nandamuri

SHARE

Figure 4. In this chart illustrating average throughput by data set size, the higher the throughput the better.

Average throughput to materialize data set into a pandas DataFrame by data size, inside and outside of Snowflake, on a CPU_X64_M machine. Small data volumes, such as the 90 MB (.09 GB) data set, have similar throughput performance, the equivalent of less than two seconds to materialize both inside and outside of Snowflake. **However, Snowflake's Container Runtime performance for the 81 GB data set is far superior.** At this high throughput rate, it takes Snowflake just 50 seconds to load data that would take nearly nine and a half minutes on an equivalent machine-type outside of Snowflake.

Distributed training APIs

Training in Container Runtime is designed to be easy, secure and performant with the Snowflake ML APIs. Container Runtime simplifies work for the end user by provisioning isolated compute resources with secure and managed node-to-node communication. When you use the Snowflake distributed trainers, you bring your own custom training code, and Snowflake manages the scheduling and distribution across resources, as defined by the user.

AUTHOR

Allie Fero

Sravankumar Nandamuri

Figure 5. Example of how to scale out a user-defined training function using the distributed PyTorch training API.

Under the hood, we are once again leveraging the Ray framework to facilitate the distribution of work across multiple devices in the cluster.

Figure 6. Container Runtime distributed training architecture.

Snowflake currently supports three distributed trainers: PyTorch, XGBoost and LightGBM. To compare performance for the XGBoost estimator, we generate a training data set with various sizes (range from 7-15 features, double type) and train a basic XGBoost regression model using the official recommended API. We found that running the same workload using the distributed Snowflake ML APIs is around 3x to 7x faster than directly running open source XGBoost.

AUTHOR

Allie Fero

Sravankumar Nandamuri

Figure 7. Snowflake ML APIs in Container Runtime speed up training by 3-7x compared to open source XGBoost by utilizing all four available GPUs.

Getting started with Container Runtime on Snowflake Notebooks

With Container Runtime, users have the flexibility to run ML workloads at scale in Snowflake without any data movement. To access Container Runtime, simply create a Snowflake Notebook and choose “Run on the container” as the backend option. With Container Runtime, you have direct access to a flexible, secure computing environment, with the option to leverage a set of preinstalled ML packages, or pip install any custom package of choice.

SHARE

AUTHOR

Allie Fero

Sravankumar Nandamuri

SHARE

Figure 8. Setting up Container Runtime in Snowflake Notebooks with just a few clicks.

Container Runtime is now available in public preview in all AWS commercial regions (excluding free trials). To try this new runtime, check out this quickstart: Getting Started with Snowflake Notebook Container Runtime that walks you through the experience of creating a Notebook and building a simple ML model. To learn how to leverage GPUs for model development in Snowflake, you can follow along in this more advanced quickstart: Train an XGBoost Model with GPUs using Snowflake Notebooks which includes a companion video demo: Accelerate The Training Of XGBoost Models Using GPUs.

For further details about Container Runtime or how to use this from Snowflake Notebooks, be sure to check out the documentation: Notebooks on Container Runtime for ML.

SHARE

RELATED CONTENT

2024년 08월 06일

Accelerating Hyperparameter Tuning in Snowflake ML

Hyperparameter optimization (HPO) is a frequently used machine learning technique to select the best parameter combination for a given model. Popular open source packages, such as scikit-learn, provide easy-to-use APIs...

Learn More

2024년 09월 17일

Snowflake Speeds Up Workloads With Hierarchical Selectivity Estimates

In this blog, we'll discuss Hierarchical Selectivity Estimates, a new Snowflake performance feature that improves...

See how

2024년 10월 10일

Snowflake Speeds Up Cloning of Large Databases By 3x On Average — Here's How We Did It

Have you ever had to wait hours to spin up a copy of your production...

More to follow

ENROLL IN SNOWFLAKE ACADEMY'S GEN AI AND ML SCHOOL

SIGN UP NOW

Snowflake Inc.

AUTHOR

Allie Fero

Sravankumar Nandamuri