PYTORCH CONFERENCE

TorchX: From Local Development to Kubernetes and Back

ML applications from research & development to production.

Authors: Joe Doliner, Jimmy Whitaker

Pachyderm, Inc.

E1

Introduction

The chasm between local development and production remains one of the biggest challenges in the application of machine learning. TorchX is an SDK for quickly building and deploying ML applications with various technologies. It allows users to create containerized definitions for PyTorch applications, making them reproducible and scalable. Data, however, typically remains one of the untracked variables in ML development, leading to gaps in reproducibility or agility.

Enabling iterative developments for data, and subsequently machine learning experiments, requires a new tooling. In this work, we combine Pachyderm, a data driven pipeline tool, with TorchX to:

- Version artifacts (data, models, code, etc.)
- Create containerized pipelines
- Automate distributed training jobs (TorchX + Pachyderm), and
- Enable incrementality

Versioning Binary Artifacts

Reproducibility in ML requires tracking how data is combined with code and any artifacts produced in the process.

Most artifacts used and produced in machine learning are binary in nature, having no predefined structure. Ensuring reproducibility requires version control and tracking lineage for all artifacts used and produced throughout the data and model development processes. Versioning binary files, requires tracking chunks of binary data and detecting when pieces of it changes. To do this files are split into chunks, each versioned with a hashing function to ensure uniqueness. Object storage is used to hold these objects, due to its scalable nature.

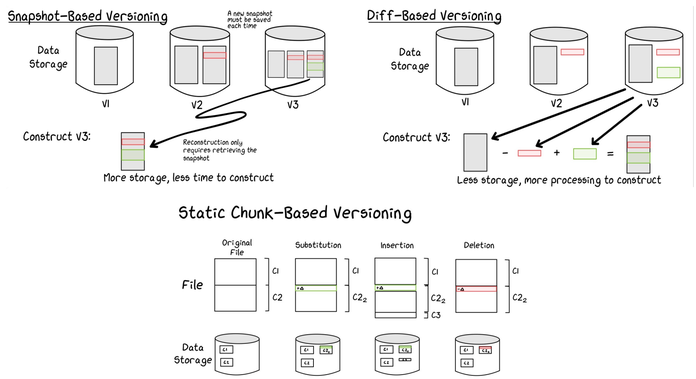

Figure 1: The data storage vs. reconstruction cost tradeoff for version control systems introduces scale and cost implications when versioning large numbers of binary artifacts.

Containerization as Pipelines

Software dependencies are packaged best with containerization.

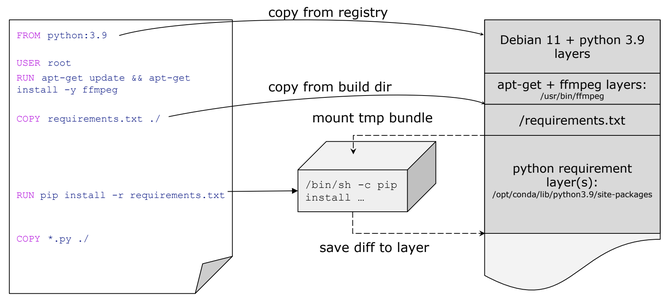

Figure 2: Docker example of containerization to capture operating system dependencies.

Dependency management is crucial for reproducible software, especially with ML code. Containerization is frequently the best way to capture all dependencies because it is able to encapsulates a whole operating system, not just python packages. This is especially important when using underlying system libraries for python libraries, for example OpenCV (Figure 2).

Reproducibility: Data Versioning with Pipelines

Reproducibility can be achieved by combining versioned data with reproducible pipelines.

Pachyderm, a system built specifically for data versioning and data triggered pipelines, can be used in conjunction with TorchX (inside versioned containers) to initiate distributed training workload, data processing pipelines, and other PyTorch components. The system provides a full lineage and history of any changes in the system, whether to the code or data.

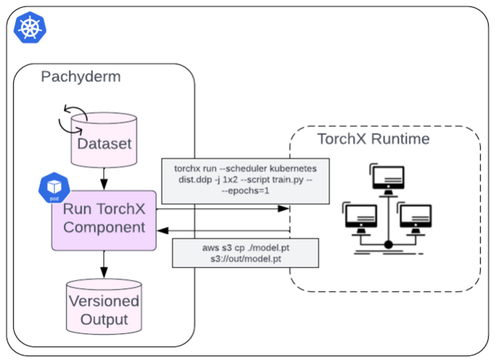

Figure 3: When the dataset is changed, a pipeline is triggered to run a TorchX job. Results from the job can be stored in the pipeline's output data repository, maintaining the provenance for the input, job execution, and any output(s).

Incrementality via Diffed Computation

Versioning enables caching previous computation.

An additional benefit derived from versioned data sources is the ability to perform incrementality, only processing what has changed. This has the benefit of being able to use only the data that has changed to update a model, rather than train from scratch each time.

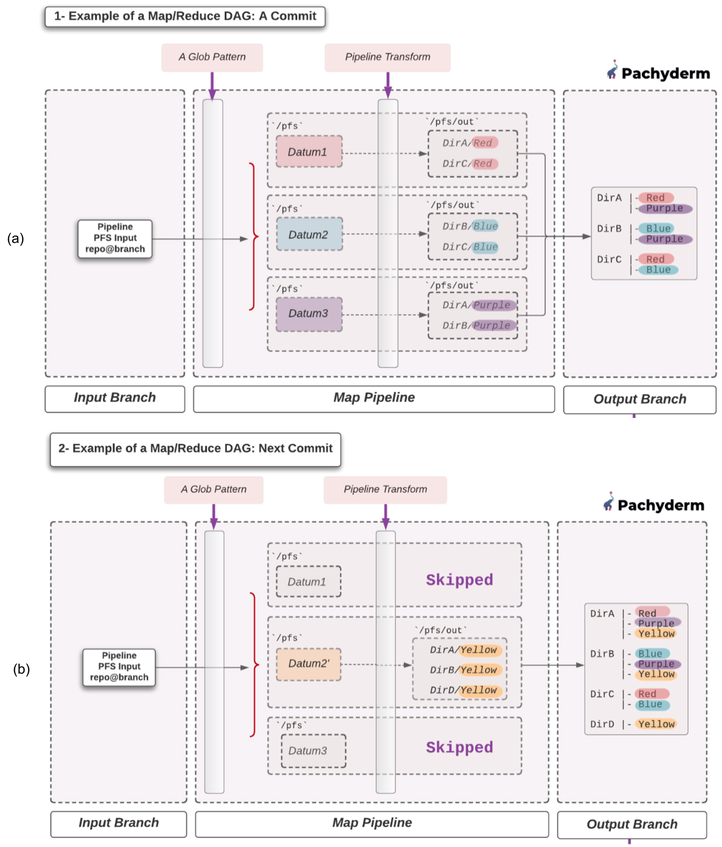

Figure 4: Pipeline incrementality in Pachyderm. (a) The default behavior for processing the data contained in the first commit is to split the data according to the parallelism pattern (glob pattern), which results in 3 parallel datums. When the data contained in datum 2 is modified (b) a job is scheduled to process that datum, while the others are skipped because they are unchanged.

References

"Pachyderm | Data-driven pipelines." https://www.pachyderm.com. (Accessed on 11/07/2022).

Velichko, I. (2022, May 24). How Docker Build Command Works Internally. iximiuz.com. https://iximiuz.com/en/posts/you-need-containers-to-build-an-image/

Take a photo to learn more: