Leveraging Large Language Models for the Automated Migration of Customer Applications
Christian Adriano, Lasse Kohlmeyer, Felix Boelter, Holger Karl, Holger Giese,
System Analysis & Modeling / Network Softwarization

Project partners

Hasso Plattner Institut
Digital Engineering · Universität Potsdam

Leveraging Large Language Models for the Automated Migration of Customer Applications

Context

Thousands of customers leverage Databricks Connect [1] to connect easily to a remote Databricks cluster and execute their Spark code written locally in an IDE of their choice. This flexibility makes it possible to integrate with the Databricks platform in any development environment or other applications. Databricks Connect is built upon Spark Connect [2] , an open source communication protocol for Apache Spark that leverages the concept of unresolved logical plans to communicate with Spark. A very large fraction of the user surface of Apache Spark is represented using this interface.

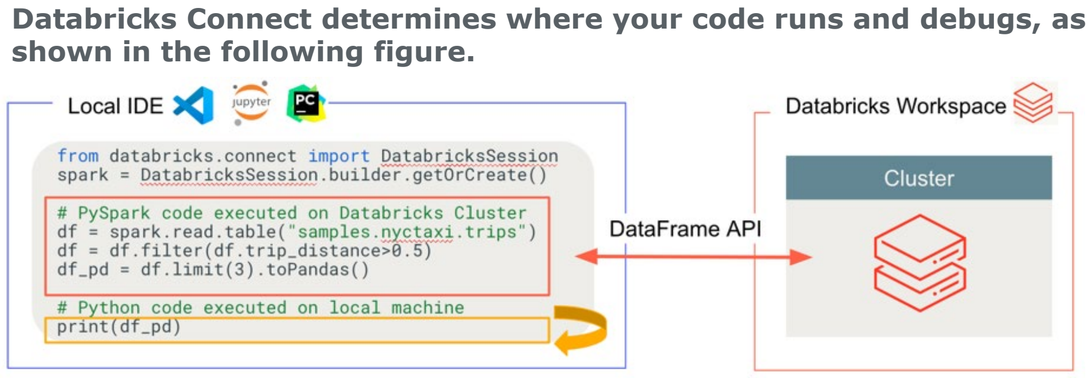
[1] What is Databricks Connect? - https://docs.databricks.com/en/dev-tools/databricks-connect/index.html
[2] Spark Connect - https://spark.apache.org/docs/latest/spark-connect-overview.html

What is Databricks connect?

Databricks Connect is a client library for the Databricks Runtime. It allows you to write code using Spark APIs and run them remotely on a Databricks cluster instead of in the local Spark session.

For example, when you run the DataFrame command `spark.read.format(...).load(...).groupBy(...).agg(...).show()` using Databricks Connect, the logical representation of the command is sent to the Spark server running in Databricks for execution on the remote cluster.

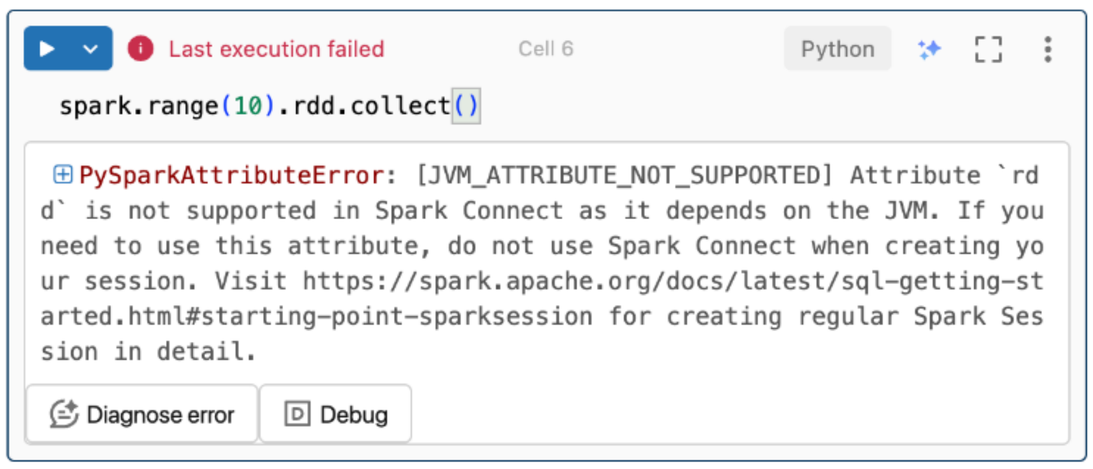
Databricks Connect determines where your code runs and debugs, as shown in the following figure.



For running code: All code runs locally, while all code involving DataFrame operations runs on the cluster in the remote Databricks workspace and run responses are sent back to the local caller.
For debugging code: All code is debugged locally, while all Spark code continues to run on the cluster in the remote Databricks workspace. The core Spark engine code cannot be debugged directly from the client.

Motivation

When we introduced Spark Connect [2] for Apache Spark, we knew that there were certain areas of the application surface that were no longer accessible because they either required direct access to the driver JVM using Py4J or required arbitrary code serialization (e.g. RDDs).



In the adoption journey of Spark Connect in the context of Databricks and Open Source we see that frequently, customers try out code that they found from internet sources or that nowadays is generated by LLMs like **ChatGPT**. For the vast majority of the existing code, there are good ways to rewrite this code using appropriate primitives that are part of Apache Spark and the DataFrame API.

Project Goals

In this Master's project, we propose to explore the following avenues to guide customers in their journey towards migrating code to be Spark Connect compatible and thus future-proof.

1. **Code analysis using LLMs** - Can they be leveraged to understand what code is compatible and how. Can they automatically rewrite code without executing it first?
2. **Exception-based migration** - Given a piece of code and an exception message, can we leverage LLMs to be able to migrate the code appropriately (and automatically write tests for it)
3. **Dynamic migration** - Based on the assumption that we know the part of the application surface that is not working, can we dynamically intercept the call path in dynamic languages like Python and rewrite the code on the fly.
4. For all of these approaches, it would be great to explore if a general-purpose LLM like **ChatGPT** is good enough or if specific fine-tuning increases the quality of the proposed changes

Tools and Processes

The team participants will decide together with the project partner the following:

- the software process (ideally a customization of the Agile Methodology for software migration)
- the acceptance criteria for "ready" and "done" tasks
- the detailed scope and its prioritization across sprints
- the development tools (IDEs, automated tools, machine learning libraries, CI/DevOps framework, etc.)

For more information contact

Christian Adriano (christian.adriano@hpi.de or Room A-2.7)