Modern Data Engineering for Data Science Projects

Project 1: (2-3 persons)
[SQL engine, query plans, engineering flavor]
Automatic query rewrite of complex SQL queries containing user-defined functions

- In:
  - plans: postgreqs / monetdb / sqlite / vertica / duckdb
- Out:
  - sql statement
- Data:
  - Tpc-h, q6 and q19
  - Text analysis
  - Zillow
  - Flights
- Steps:
  - Convert plan into a data flow graph (DFG)
  - Generate SQL from a DFG

Project 2:   (1-2 person)
[algorithmic, graph traversal, dynamic programming]
Operator fusion using dependency analysis

- In:
  - Data flow graph representing a query plan
- Out:
  - Fused query plan
- Data:
  - DFG graphs for zillow vs all, flights
- Steps:
  - Identify sets of operators that can be fused
  - Create fused query plan following a DP, recursive approach

Project 3:   (1-2 persons)
[query optimization, machine learning]
Learning cost models for black-box operators using Bayesian Optimization

- In:
  - An operator, a set of platforms, execution parameters
- Out:
  - Cost model for the operator: statistics estimates, cost function
- Data:
  - Platforms: Monetdb, PostgreSQL
  - Queries: tpc-h, zillow, flights, textmining
- Steps:
  - Learn/read about BO, read cherrypick paper[1]
  - Setup testbed

Project 4: (2 persons)
[familiarity with Python booster technology: transpilers, IR]
Convert Python programs to SQL queries

- In:
  - A set of python programs
- Out:
  - The corresponding SQL statements
- Data:
  - Platforms: SQLite, Monetdb, PostgreSQL
  - Queries: tpc-h, zillow, flights, text-mining (focus on 4-5 udf's)
- Steps:
  - Use a Python-to-SQL transpiler (e.g., Grizzly) to parse Python programs
  - Use an IR-based approach (e.g, Weld) to parse Python programs
  - Extend the mechanism to support user-defined functions

Project 5: (2-3 persons)
[familiarity with big data platforms, develop analytical skills, engineering flavor]
Benchmarking big data systems for data science queries

- In
  - SQL with Python UDFs
- Out
  - Experimental analysis
- Data
  - Platforms: SQLite, DuckDB, PostgreSQL, MonetDB, Vertica (community edition), Spark, MongoDB, Dask
  - Datasets: zillow, flights, logs, tpch, 311
  - Dimensions: parallelism (single-threaded vs multi-threaded), caches (hot/cold), data size
- Steps
  - Investigate and deploy platforms
  - Run experiments

Project 6: (1-2 persons)
[programming puzzle]
Support Dynamic typed Python UDFs in databases

- In
  - Dynamic typed Python UDF
  - SQL query
- Out
  - Python UDF is registered and query runs
- Data
  - Platforms: PostgreSQL, MonetDB

    - Datasets: zillow, flights
- Steps
    - Investigate input data types via query plan
    - Investigate output data types via sampling and sql extensions
    - Wrap the python function with a create function statement
    - Rewrite and execute the SQL query

References and software links

[1] Omid Alipourfard, Hongqiang Harry Liu, Jianshu Chen, Shivaram Venkataraman, Minlan Yu, and Ming Zhang. 2017. CherryPick: Adaptively Unearthing the Best Cloud Configurations for Big Data Analytics. In 14th USENIX Symposium on Networked Systems Design and Implementation (NSDI 17). 469–482

[2] Palkar, S., Thomas, J. J., Shanbhag, A., Narayanan, D., Pirk, H., Schwarzkopf, M., ... & Zaharia, M. (2017). Weld: A common runtime for high performance data analytics.

[3] Hagedorn, Stefan, Steffen Kläbe, and Kai-Uwe Sattler. "Putting Pandas in a Box." *CIDR*. 2021.

[2] https://github.com/weld-project/weld

[4] https://github.com/dbis-ilm/grizzly