List of Project Topics (Proposals)

Last update: Oct 20, 2023

LDE Projects in the Context of Two Open-source Systems

- **DAPHNE EU-project**
  https://github.com/daphne-eu/daphne
  - Focus on integrated data analysis pipelines
  - Project implementation in C++ and Python

- **Apache SystemDS**
  https://github.com/apache/systemds
  - Focus on the end-to-end data science lifecycle
  - Project implementation in Java, Python, and DML

Topics in DAPHNE

Most projects in C++, some partly in Python

#627: Optimizing Code Generation for Matrix Multiplication

- **Motivation**
  - General matrix multiplication (GEMM) is a core operation in many ML algorithms, and also one of the heavy-hitters in regards to the runtime performance of these algorithms.
  - DAPHNE provides a highly optimized implementation of the GEMM operator through the openBLAS library.
  - Recently, a direct generation of the code for matrix multiplication was added.
  - However, the currently generated code for the GEMM operator is not competitive in terms of runtime performance with the openBLAS implementation.
- **Task** (in C++)
  - Implement the generation of more optimized matrix multiplication code in the DAPHNE compiler, including optimizations such as tiling, vectorization, and parallelization.
- **More information & hints**
  - https://github.com/daphne-eu/daphne/issues/627
  - Contact: Philipp Ortner

#628: Efficient Processing of Star Schema Benchmark

- **Motivation**
  - Relational query processing is an integral part of integrated data analysis pipelines, but so far has limited support in DAPHNE
- **Task** (in C++)
  - Take the well-known Star Schema Benchmark (SSB) as an example and improve/extend the required system components of DAPHNE to make it work.
  - Add and/or improve the required components
    - SQL parser
    - Optimizer/compiler (e.g., relational algebra rewrites, operator ordering, pipeline fusion)
    - Runtime/execution engine (e.g., efficient physical operators/kernels for joins etc., integration with DAPHNE's vectorized execution engine)
  - Goal: Performance competitive to existing systems like DuckDB, MonetDB, pandas
- **More information & hints**
  - https://github.com/daphne-eu/daphne/issues/628
  - Contact: Patrick Damme

#629: Processing of String Data Sets

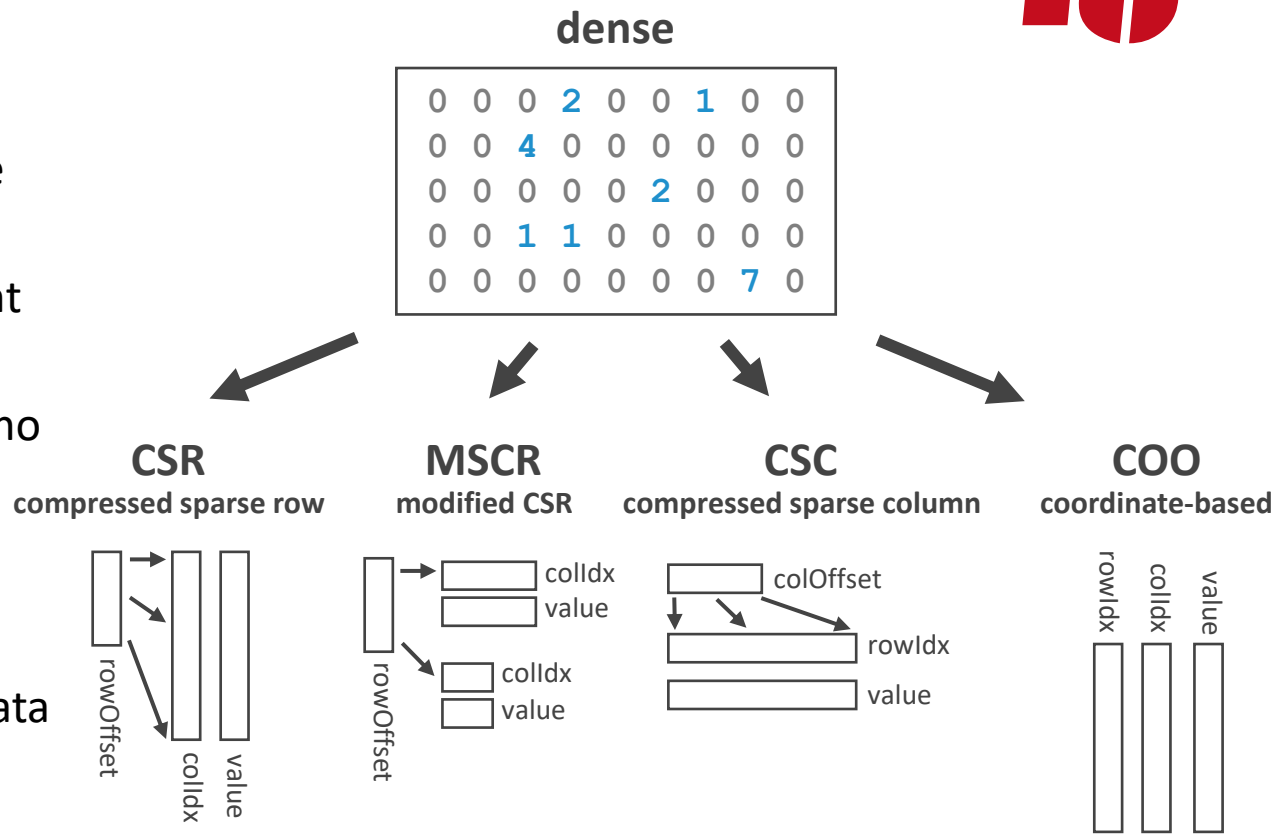
- **Motivation**
  - Many real-world data sets contain string-valued attributes,
    but so far there is limited support for strings in DAPHNE.
- **Task** (in C++)
  - Given a data set (file on storage or pandas.DataFrame in memory) with string columns.
  - Read the file or transfer the data frame to DAPHNE via shared memory.
  - Perform some simple feature transformations (e.g., one-hot, dictionary coding, to be implemented)
    on the data and feed it into given ML algorithms and/or perform SQL queries on the data set.
  - Implement and/or improve the required system components
    (especially string representation, string processing, and string I/O)
- **More information & hints**
  - https://github.com/daphne-eu/daphne/issues/629
  - Contact: Patrick Damme

#500 End-to-end Sparsity Exploitation in DaphneDSL

- **Motivation**
  - Sparse matrices (most cells are zero) are commonplace in data science applications
  - DAPHNE supports a CSR (compressed sparse row) representation, but support is still a proof-of-concept
- **Task** (in C++)
  - Enable end-to-end sparsity exploitation (sparsity estimation, sparse kernels, dense/sparse selection, integration into DAPHNE's vectorized engine)
- **More information & hints**
  - https://github.com/daphne-eu/daphne/issues/500
  - Contact: Patrick Damme

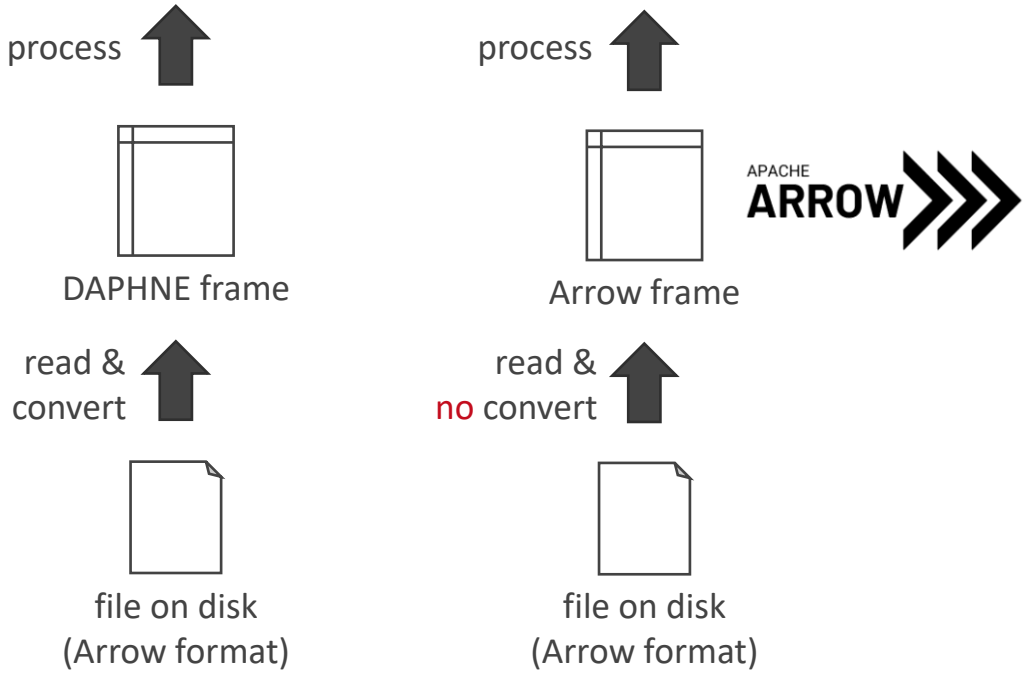
#528 Additional Sparse Matrix Representations: COO

- **Motivation**
  - Sparse matrices (most cells zero) are commonplace in data science applications
  - Various sparse matrix representations with different characteristics, pros, and cons
  - So far, DAPHNE supports CSR, MCSR, and CSC, but no suitable representation for ultra-sparse matrices
- **Task** (in C++)
  - Extend DAPHNE by additional sparse matrix data type for COO, which can be used for ultra-sparse data
  - Add kernels (physical operators) with efficient algorithms specialized for this sparse format
- **More information & hints**
  - https://github.com/daphne-eu/daphne/issues/528
  - Contact: Patrick Damme



#502 Apache Arrow for Data Frames in DAPHNE

- **Motivation**
  - DAPHNE supports relational operations on frames, and offers a column-major frame representation
  - Apache Arrow is a widely used framework and exchange format for columnar data; supporting its memory layout in DAPHNE would improve the interoperability with other libraries and tools
  - In particular, reading Arrow files could become more efficient by avoiding unnecessary conversions
- **Task** (in C++)
  - Extend DAPHNE by a new frame data type which uses the memory layout of Apache Arrow
  - Add kernels for a few relational operations on the Arrow-based frame representation
  - Add a file reader for Arrow avoiding conversions



- **More information & hints**
  - https://github.com/daphne-eu/daphne/issues/502
  - Contact: Patrick Damme

Topics in Apache SystemDS

Most projects either in Java or in DML (SystemDS's R-like domain-specific language), some partly in Python

#3178: Tuple Deduplication

- **Motivation**
  - Deduplication is an important data cleaning/preprocessing step
- **Task** (in DML)
  - Implement a dedup() built-in function in SystemDS
  - Challenge: Make pairwise comparison efficient (e.g., clustering/blocking, hashing, sorting)
- **More information & hints**
  - https://issues.apache.org/jira/browse/SYSTEMDS-3178
  - Contact: Matthias Boehm

#3549: Hidden Markov Model Built-in Function

- **Task** (in DML)
  - Hidden Markov models are a classic machine learning algorithm.
  - This task is to implement a Hidden Markov Model the DML language of SystemDS.
  - The implementation should be vectorized and avoid index lookups as much as possible.
  - See also: https://en.wikipedia.org/wiki/Hidden_Markov_model
  - In specific, we would be interested in an implementation that, given k last timesteps, predicts the next event.
  - The implementation should input a matrix with a time series running down through columns, and then train a model that is able to predict the next x events.
  - As a result two builtins are expected: HMM and HMMPredict.

- **More information & hints**
  - https://issues.apache.org/jira/browse/SYSTEMDS-3549
  - Contact: Matthias Boehm

#3553: Additional DNN/Factorization Optimizers and Preconditioners
#3259: NN Builtin: Add Shampoo Optimizer

- **Task** (in DML)
  - AdamW (Adam with weight decay)
  - Shampoo
    - Shampoo paper: https://arxiv.org/abs/1802.09568
    - Add the optimizer similar to existing NN optimizers like SGD, SGD_nesterov, RMSProp, Adam
  - ScaledGD
- **More information & hints**
  - https://issues.apache.org/jira/browse/SYSTEMDS-3553
  - https://issues.apache.org/jira/browse/SYSTEMDS-3259
  - Contact: Matthias Boehm

#3535 Implement Selected TPCx-AI Use Cases

- **Task** (in DML, Java)
  - Implement SLAB (Scalable LA Benchmark)
  - Implement Selected MLPerf Use Cases
- **More information & hints**
  - https://issues.apache.org/jira/browse/SYSTEMDS-3535
  - Contact: Matthias Boehm

#3213 Builtin for Cluster-based Quantization

- **Task** (in DML)
  - DML-based built-in function for quantization of numerical vectors via clustering as described as product quantization (https://openaccess.thecvf.com/content_cvpr_2013/papers/Ge_Optimized_Product_Quantization_2013_CVPR_paper.pdf).
  - Here, vectors (rows) are split into sub-vectors and ran through K-Means clustering and the cluster id is used as codeword for each subvector.
- **More information & hints**
  - https://issues.apache.org/jira/browse/SYSTEMDS-3213
  - Contact: Matthias Boehm

Fourier Transformations

- **Task** (in DML)
  - Add built-in support for Fast Fourier transformation (FFT) and Short time Fourier transformations (STFT).
  - An example of TensorFlow using Fourier transformations for audio input: https://www.tensorflow.org/tutorials/audio/simple_audio
  - The scope of this project is limited to the construction of the Fourier transformations and is not intended to extend into the construction of a neural network.
- **More information & hints**
  - Contact: Sebastian Baunsgaard, Carlos E. Muniz Cuza

Non-linear Dimensionality Reduction

- **Task** (in DML)
  - Implement new built-in algorithms for nonlinear dimensionality reduction.
  - Algorithms to build/choose from and evaluate include
    - t-distributed stochastic neighbor embedding (t-SNE)
    - uniform manifold approximation and projection (UMAP)
    - potential of Heat-diffusion for Affinity-based Transition Embeddings (PHATE).
- **More information & hints**
  - Contact: Sebastian Baunsgaard

Additional Compression Formats for Matrix Column Groups

- **#3539: Delta Encoding** (in Java)
  - Reader for uncompressed matrices to be encoded into compression statistics as if delta-encoded
  - Transforming operations such as cumulative sum that have to be wired to transform existing column groups into a delta-encoded column group
  - Compression taking an uncompressed matrix and encoding a delta-encoded column group from it without materializing the delta encoded version of the input matrix
- **#3543: Piece-wise Linear Compression** (in Java)
  - Implement a new column group for piece-wise linear compression that is based on a target loss
  - The technique compresses a column of values, into smaller line segments
  - A naive implementation of this in the extreme cases would potentially be 0 target loss, with full allocation of input, and 100% target loss containing only the average of all values
  - Other than this, the implementation moves from a lossless array into a lossless piece-wise linear representation via dynamic programming.
- **More information & hints**
  - https://issues.apache.org/jira/browse/SYSTEMDS-3539
  - https://issues.apache.org/jira/browse/SYSTEMDS-3543
  - Contact: Sebastian Baunsgaard

#3540: Compression: Specialized Co-coding Algorithm

- **Task** (in Java)
  - A new technique for co-coding columns that searches for larger groups of columns to combine rather than singular groups
  - Specifically, we are looking for ways to detect instances of multiple columns that have the same number of unique values, and co-code perfectly together since they
    - 1. either are one hot encoded, or
    - 2. perfectly map each unique value to the same unique value of the other
  - In other cases we need to fallback to our default encoding
- **More information & hints**
  - https://issues.apache.org/jira/browse/SYSTEMDS-3540
  - Contact: Sebastian Baunsgaard

#3541: Exploratory Workload-aware Compression on Intermediates

- **Task** (in Java)
  - This project is to experiment with enabling compression on intermediate values.
  - Currently, the project supports compression of arbitrary intermediate values.
  - The goal of this project is to enable this feature, experiment with it across a number of algorithms, and report results, bugs and interesting findings.
  - In this process, if regressions or limitations are found, solutions are proposed and implemented.
- **More information & hints**
  - https://issues.apache.org/jira/browse/SYSTEMDS-3541
  - Contact: Sebastian Baunsgaard

#3548: Optimize I/O Path of SystemDS Python Interface

- **Task** (in Java and Python)
  - This task is to optimize the data transfer between SystemDS and Python
  - Two starting points are string transfer of pandas data frame data both to and from SystemDS and boolean transfer from SystemDS that could be bit packed
  - The task includes benchmarking the interface to know how the performance is currently and what the limiting factors are
  - There is also an opportunity to try out parallel data transfer between the environments
  - There are multiple low hanging fruits for this task:
    - String transfer – This is currently done by transferring a single string at a time, making it unbearably slow because it calls Py4J once per cell.
    - Bit packed transfer – This is currently done by unpacking bits from a long into individual bytes, making the transfer 8x larger than it is supposed to be.
- **More Information & Hints**
  - https://issues.apache.org/jira/browse/SYSTEMDS-3548
  - Contact: Sebastian Baunsgaard

Federated/Distributed Frame Operations

- **Motivation**
  - Some of the recently added frame operations (e.g., detect schema (column types), removeEmpty (removes empty rows), etc.) are currently only supported for local execution in the control program (CP)
  - Our group investigated the parallelization of feature transformations on frames in a recent paper; however, so far only multi-threading on a local CPU is supported for the parallelization
- **Task** (in Java and DML)
  - This project extends upon that previous work by enabling the parallelization of frame operations in distributed and federated environments
- **More information & hints**
  - See our recent paper: *Arnab Phani, Lukas Erlbacher, Matthias Boehm: UPLIFT: Parallelization Strategies for Feature Transformations in Machine Learning Workloads. Proc. VLDB Endow. 15(11): 2929-2938 (2022)* https://www.vldb.org/pvldb/vol15/p2929-phani.pdf
  - Contact: Matthias Boehm

Additional GPU Kernels

- **Motivation**
  - For hardware acceleration, SystemDS already supports many operations on GPUs
  - However, several operations are still missing
- **Task** (in Java and C++/CUDA)
  - Implement additional GPU kernels for more operations and/or for sparse matrices, examples include:
    - removeEmpty (removes all-zero rows in matrix, challenging on GPUs since the output size depends on the data; cumulative sum can come in handy, but its parallelization is challenging)
    - set indexing (given a set of non-negative ints, extract the rows/columns at these positions from a matrix)
    - more operations on sparse matrices (most of the current GPU kernels work on dense matrices, sparse is much more challenging on GPUs)
    - rand (generation of a matrix of random values, the special challenge is to ensure the same output as the local, single-threaded, CPU variant if a certain seed is given; for that reason one cannot simply use a vendor-provided random number generation for the GPU)
- **More information & hints**
  - Contact: Matthias Boehm

#3534 Extended Feature Transformations

- **Task** (in Java)
  - Implement various rewrites and optimizations for the current multi-threaded feature transformations and in general for feature engineering tasks.
  - Task-level reuse in feature engineering: Every feature transformation is split into two tasks - Build and Apply. The Build tasks produce the dictionaries and the Apply tasks use those dictionaries to encode the features. A typical feature engineering use case explores multiple combinations of feature transformations (e.g. recoding vs feature hashing vs dummy coding), which opens up opportunities to build the dictionaries once and reuse them for other transformations. The project is to build a cache for the partial intermediates of the tasks and reuse if possible
  - Fuse feature transformation tasks: Build and Apply phases of different feature transformation tasks can be fused to save computation and memory usage. For example, Recode Build and Apply can be done together. This project is to identify these cases and implement them.
- **More information & hints**
  - https://issues.apache.org/jira/browse/SYSTEMDS-3534
  - Contact: Arnab Phani

#3333 Extended Matrix Multiplication Chain Optimization

- **Task** (in Java)
  - This task aims to extend the existing matrix multiplication chain optimization (org/apache/sysds/hops/rewrite/RewriteMatrixMultChainOptimization) in multiple ways. First, the rewrite should consider transpose and element-wise *, +, /, -, which otherwise would reduce the scope of optimization. Second, mmchain optimization currently only optimizes chains of matrix multiplications. Instead, the extension should optimize DAGs (directed acyclic graphs) where common an input matrix or matrix multiplication result might be used multiple times in the chain.
- **More information & hints**
  - https://issues.apache.org/jira/browse/SYSTEMDS-3333
  - Contact: Matthias Boehm

Image Processing in SystemDS

- **Compound image transformations** (in Java)
  - Combine multiple image transformations into a single transformation matrix in linear algebra.
  - Using the combined matrix, construct a (most likely sparse) matrix that, when multiplied with the linearized input matrix, constructs the transformed image.
  - The matrix construction instruction should have argument support for different interpolation algorithms, e.g. Nearest Neighbor, Linear Interpolation, or Cubic interpolation.
  - See also: https://en.wikipedia.org/wiki/Affine_transformation
- **Image IO** (in Java)
  - Construct new IO read and write operations that can
    - read all images contained in a folder as linearized matrix rows
    - write a matrix to a folder where each row is saved as an image on disk
  - Use lossless image formats to write to disk, but optionally this task can add support for lossy image formats.
  - All images read and written need to have the same number of row and column pixels.
- **More information & hints**
  - Contact: Sebastian Baunsgaard

Execution Plan Visualizer for Machine Learning Workloads

- **Task** (programming language up to discussion, e.g., Python)
  - A typical machine learning (ML) system compiles a high-level script into hybrid exection plans of local, distributed and GPU operations. Understanding these plans are absolutely necessary to implement features in the compilation and runtime stacks. This project aims to extend the current SystemDS DAG visualizer by integrating runtime plans, stat outputs and lineage traces, as well as improve the user interfaces. The current visualizer only supports the compile-time plans. SystemDS recompiles parts of the plan in runtime and updates the operators accordingly. The stat output shows the time taken by individual operations, and the lineage trace provides the sequence of executed operations. This project combines these logs and visualizes the execution plans in different timeframe (compile-time, runtime and post execution).
- **More information & hints**
  - Contact: Arnab Phani

#3550: Heuristic-based Transformation Spec Generator

- **Task** (in Java)
  - This project is to define a new method that heuristically generate transformencode specifications based on the metadata of input frames.
  - The project is beneficial in AutoML scenarios where we do not know what the best way of encoding our non-numeric inputs into numeric values is.
- **More information & hints**
  - https://issues.apache.org/jira/browse/SYSTEMDS-3550
  - Contact: Arnab Phani

More Topics in Apache SystemDS

- **Loop Vectorization** (in Java)
- **Extended Common Subexpression Elimination** (in Java)
- **Extended I/O Framework: Readers/Writers for More File Formats (NetCDF, HDF5, Arrow)** (in Java)
- **#3552 Additional SparseBlock Layout: Double-Compressed Sparse Row (DCSR)** (in Java)
  - https://issues.apache.org/jira/browse/SYSTEMDS-3552

- **More information & hints**
  - Contact: Matthias Boehm

Stand-alone and Cross-cutting Topics

#3191 Entity Resolution for Plagiarism Detection

- **Task** (in DML)
  - As an exploratory analysis, it would be very interesting to build an example entity resolution pipeline for plagiarism checks of submitted code projects. At the core of this pipeline, we can leverage recent work on embeddings for computing code similarity (https://arxiv.org/pdf/2006.05265.pdf) and then wrap this into an end-to-end pipeline. After an initial implementation outside SystemDS, we should investigate the missing features to run it end-to-end in SystemDS as well.
- **More information & hints**
  - https://issues.apache.org/jira/browse/SYSTEMDS-3191
  - Contact: Matthias Boehm

Reuse of Intermediates in Exploratory Data Science Pipelines

- **Task** (in Python)
  - Data science tasks such as feature engineering, data cleaning, hyperparameter tuning and network architecture search are exploratory and hierarchically composed. Data scientists iterate on data science pipelines by updating each task in the pipeline until the desired accuracy is achieved. This practice leads to redundant operations. For example, a pipeline for sentiment analysis on news headlines may include tokenizing, embedding, feature representation and training. The data scientists will try different tokenizers (n-gram, sequence), TF-IDF for feature representation and different models (logistic regression, neural network) iteratively to find the best pipeline. This workflow has many possible redundancies (e.g., trying a different model requires the same data pre-processing). Previous work addressed this redundancy by coarse-grained reuse, multi-query optimization, and fine-grained reuse inside ML systems. The goal of this project is to optimize these pipelines by reusing previously executed operations. Most ML pipelines utilizes multiple libraries (e.g., Spark, TensorFlow) and are writen in Python. This project will first extract the abstract syntax trees from the pipeline using Python's ast module, explore the possibility to use the AST as a key to an operation and implement a cache to identify and reuse these operations.
- **More information and hints**
  - Contact: Arnab Phani

TerseTS: A Package for Time Series Compression with Python

- **Task** (in Python)
  - The idea of the project is to implement a Python package containing a variety of primitives for lossless and lossy time series compression methods. Currently, such a package is non-existent and developers have to implement their compression primitives all the time from scratch or search the web for individual open-source implementations.
  - The package could be made public and uploaded to the Python Package Index (PIP), where developers can have access.

- **More information & hints**
  - Contact: Carlos E. Muniz Cuza

Testing SystemDS and DAPHNE with DSL Fuzzing

- **Task** (programming language up to discussion, e.g., Python)
  - Implement a framework for generating source code for sequences of linear algebra, that can be executed in different systems/frameworks to compare
    - semantic equivalence
    - performance between the frameworks
- **More information & hints**
  - Contact: Patrick Damme, Sebastian Baunsgaard

Alternative: Propose Your Own Topic Idea

- **We are open to additional topic proposals**
  - In the context of data engineering, data management, and machine learning systems
  - If you are passionate about your idea
  - More topics in SystemsDS and DAPHNE
  - Other open-source systems possible, but contributions might be more difficult to get accepted