

List of Project Topics (Proposals)

Last update: Apr 28, 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

■ Motivation

- DAPHNE has a domain-specific language DaphneDSL for linear/relational algebra on matrices and frames, great for building complex data analysis algorithms
- BUT: Data scientists prefer working at a higher abstraction level with primitives for common analytics tasks (e.g., data cleaning, clustering, ...)
- SystemDS has a powerful library of such primitives written in its domain-specific language DML

■ Task (in Python or another suitable language)

- Make SystemDS's library of ML primitives available to DAPHNE users
- In that context: Stand-alone tool translating scripts written in SystemDS's DML to DAPHNE's DaphneDSL


 Apache SystemDS™

```
shiftAndScale = function(Matrix[Double] X)
  return (Matrix[Double] R)
{
  R = (X - rowMeans(X)) / rowSds(X);
}
```

 DAPHNE

```
def shiftAndScale(X:matrix<f64>) -> matrix<f64> {
  return (X - mean(X, 1)) / stddev(X, 1);
}
```

automatic
translation



■ More information & hints

- <https://github.com/daphne-eu/daphne/issues/497>

#498 Update-in-place Operations

Taken by a master student, additional team members (master students) possible

Motivation

- DAPHNE maps logical ops from linear/relational algebra to calls to kernels (C++ functions consuming and producing DAPHNE matrices/frames)
- Kernels always create a new data object for their output, causing additional memory accesses
- *Under certain conditions*, the input data objects could be reused to improve memory consumption and performance

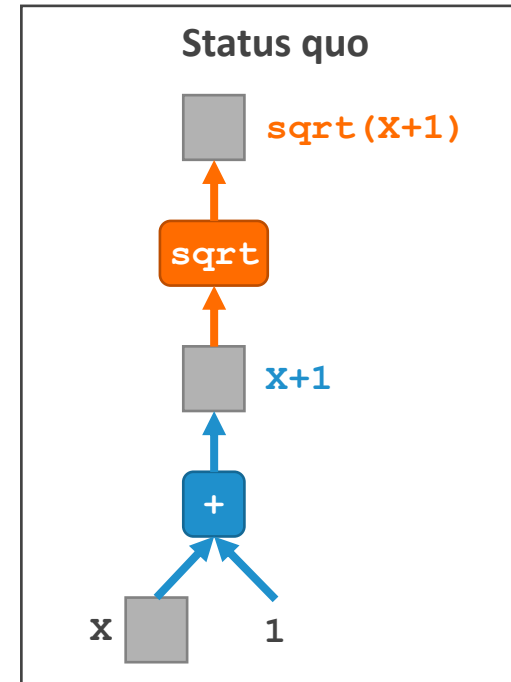
Task (in C++)

- Enable update-in-place optimizations in DAPHNE
- Concerns DAPHNE compiler, runtime infra, kernels

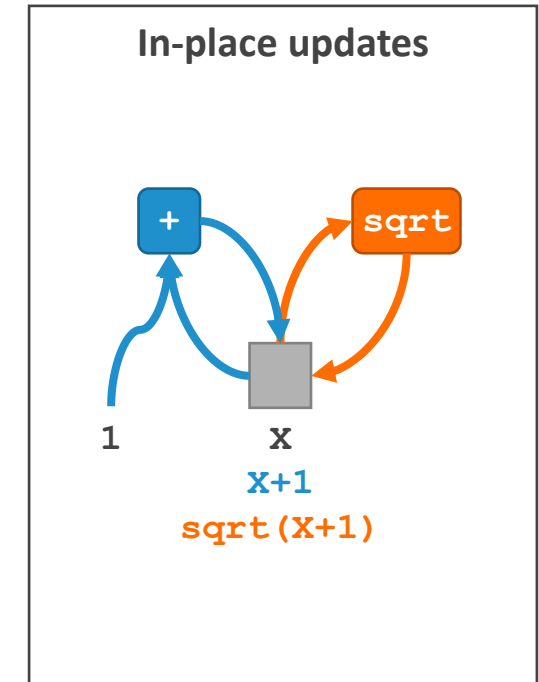
More information & hints

- <https://github.com/daphne-eu/daphne/issues/498>

example: evaluation of `sqrt(X+1)` on matrix `X`



Newly created matrix after each operation



Reusing matrix `X` for both ops

#499 Connecting DAPHNE to the Data Science Ecosystem: Efficient Data Exchange with Popular Python Libs



■ Motivation

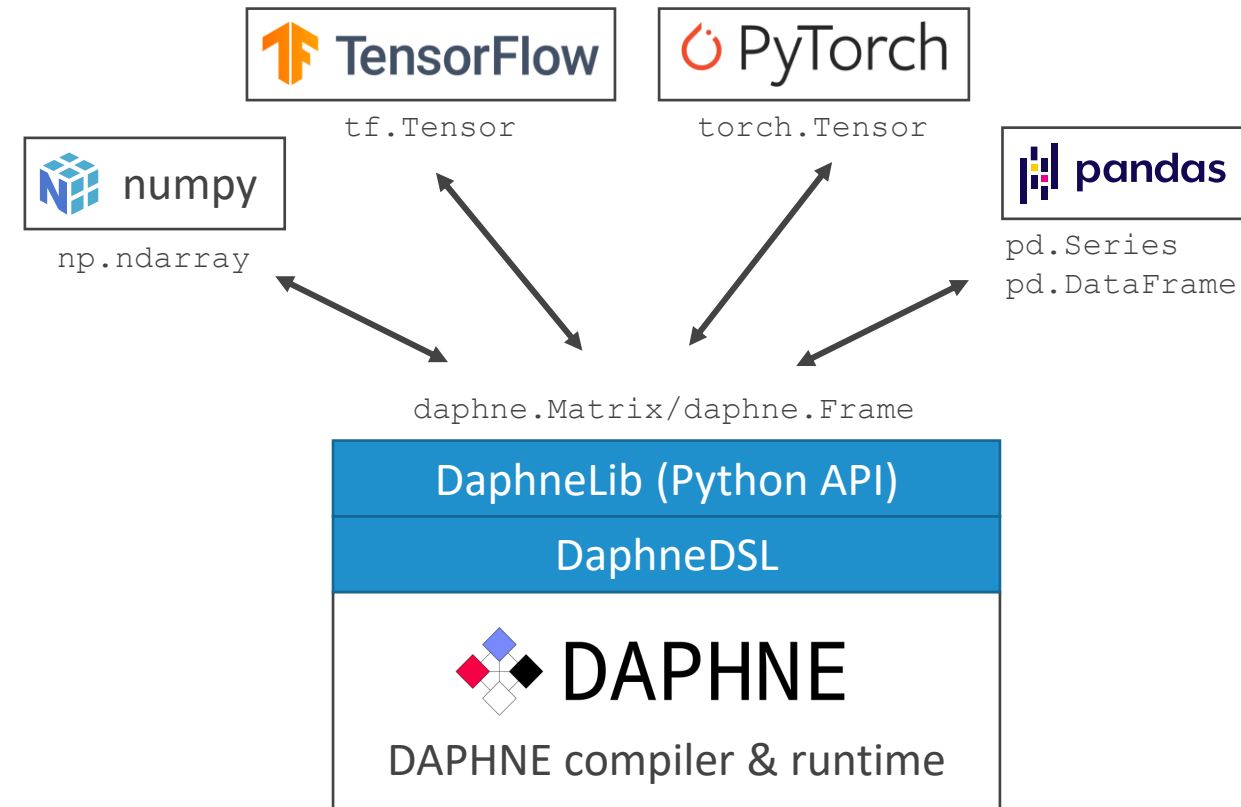
- Data scientists prefer Python nowadays
- DAPHNE has a Python API (DaphneLib)
- Efficient data exchange is required to successfully integrate DAPHNE with the data science ecosystem

■ Task (in Python and C++)

- Extend existing infra for efficient data exchange between DAPHNE and numpy by additional Python libraries such as pandas, TensorFlow, and PyTorch

■ More information & hints

- <https://github.com/daphne-eu/daphne/issues/499>



#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

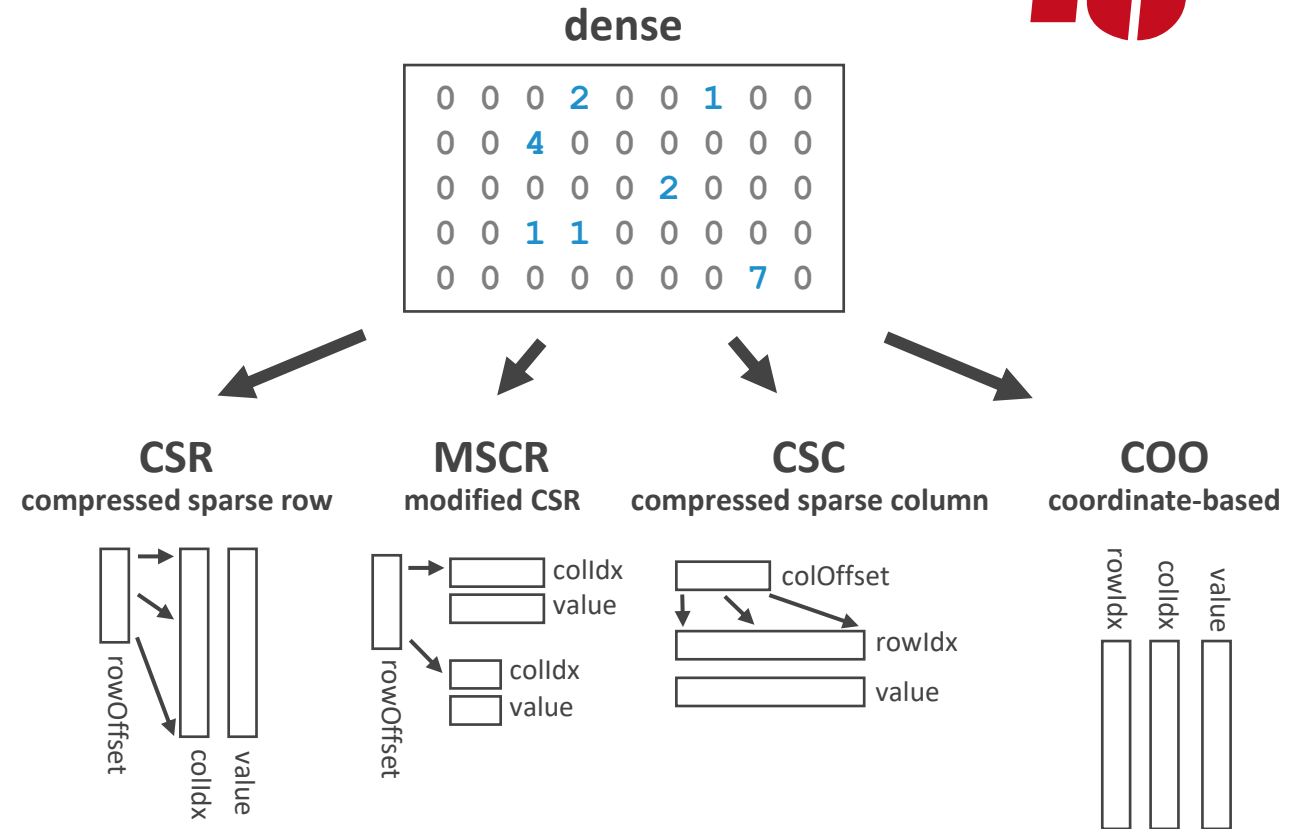
- <https://github.com/daphne-eu/daphne/issues/500>

■ 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 only CSR, would benefit from rich choice of representations to better adapt to various situations

■ Task (in C++)

- Extend DAPHNE by additional sparse matrix data types (e.g., MCSR, CSC, COO)
- Add kernels (physical operators) with efficient algorithms specialized for these sparse formats
- Devise selection of a suitable representation



■ More information & hints

- <https://github.com/daphne-eu/daphne/issues/501>

#502 Apache Arrow for Data Frames in DAPHNE

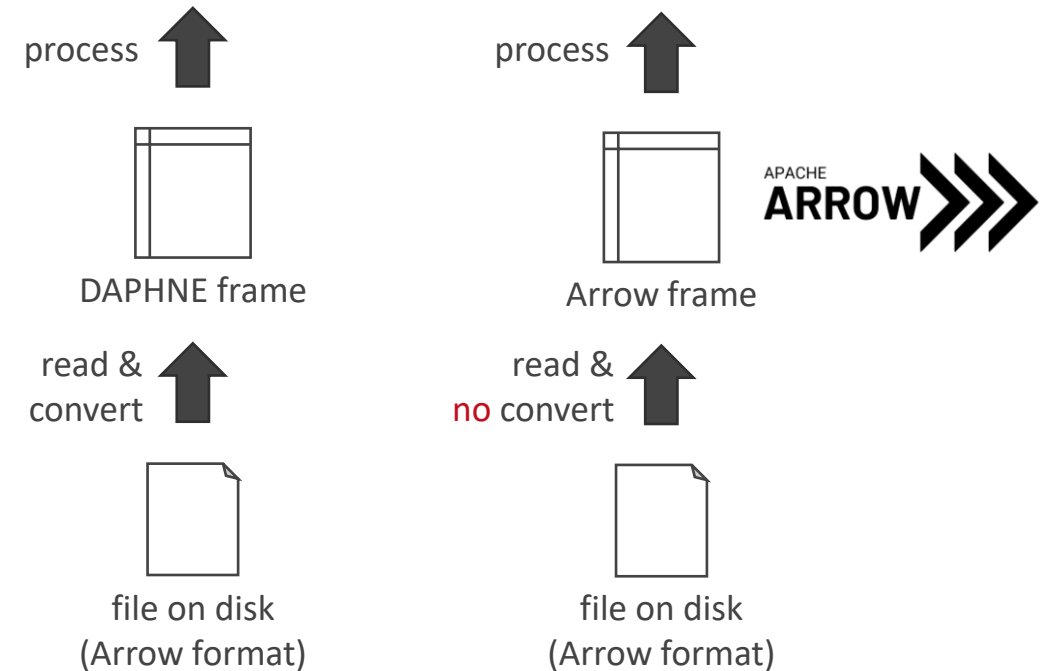


■ Motivation

- 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>

More Topics in DAPHNE



- **Improvements to Vectorized Engine and Operator Fusion**
 - Implementation in C++
- **#525 DaphneDSL map() UDFs in Other Languages (e.g., Python, C++)**
 - Implementation in C++ and the selected other languages
 - More information & hints: <https://github.com/daphne-eu/daphne/issues/525>
- **Comprehensive Test Suite (Unit Tests and Script-Level Tests, DSL Fuzzing)**
 - Implementation in C++, DaphneDSL, and potentially other languages
- **More information & hints**
 - On request

■ Motivation

- GCLs are the building blocks of many Graph Convolutional Networks (class of neural network for processing unstructured data representable as a graph)
- Applications in social networks, molecular biology, multivariate timeseries
- Key design element: pairwise message passing between neighbor nodes
- Multiple types of mechanisms for pairwise message passing

■ Task (in DML)

- Implement the most relevant GCLs in SystemDS
- Benchmark their performance on real-life data

■ More information & hints

- https://pdamme.github.io/teaching/2023_summer/lde/GraphConvolutionalLayers.pdf

DAG Visualizer for Machine Learning Workloads

■ Motivation

- A typical machine learning (ML) system compiles a high-level script into hybrid execution plans of local, distributed and GPU operations. Understanding these plans is absolutely necessary to implement features in the compilation and runtime stacks.

■ Task (preferably in Python, other languages possible on request)

- This project aims to build a toolkit that takes an execution plan of Apache SystemDS as input and produce a visual structure of the operations. A good example is TensorFlow's TensorBoard. However, unlike TensorBoard, this project aims to help the ML system researchers. The visualization may include various metadata associated with the operations such as predicted compute and memory costs, execution modes and instruction order. This visualization will allow the researchers to reason about alternative plans with multiple backends and devices. Furthermore, the toolkit should expose an API to add or remove metadata easily to support future work.

■ More information & hints

- On request

#3430: Query Interface over Lineage Traces

Taken by a bachelor student, additional team members (bachelor students) possible

- **Task** (preferably in Java, Python or other languages on request)
 - Build a query interface over many text-based lineage traces from various ML workloads. A lineage trace is a serialized DAG of the operations without any control flows. The task is to deserialize the traces into in-memory formats and answer queries regarding the workload characteristics. The in-memory format could be tabular or semi-structured. The internal representation should preserve the structure of the DAGs and the operators' properties to answer all kinds of queries. One possible way would be to represent each DAG by multiple relations – one for each operator with the corresponding attributes, and one for preserving the structure with attributes including output nodes for each operator. Existing libraries (e.g., Pandas) can be used to define the query interface.
 - Example queries include
 - Find all DAGs with a convolution operation that takes more than 20ms to execute.
 - Compare the total number of operations between two DAGs.
 - Group DAGs by the type of non-linear operation used and calculate the average execution time for each group.
 - Compare the memory usage of the matrix multiplication between two DAGs.
 - Find similar DAGs on different datasets.
- **More information & hints**
 - <https://issues.apache.org/jira/browse/SYSTEMDS-3430>

#3151/#3163: Missing Value Imputation & Amputation

■ Motivation

- Missing data values are a typical issue encountered in real-world data sets

■ Task (in DML)

- Given a data set with missing values
 - Classify the missing values as Missing Completely at Random (MCAR), Not Missing at Random (NMAR), and Missing not at Random (MNAR)
 - If NMAR: Impute the missing values
- Given a dataset without missing values: ampute a part of the values with various techniques

■ More information & hints

- <https://issues.apache.org/jira/browse/SYSTEMDS-3151>
- <https://issues.apache.org/jira/browse/SYSTEMDS-3163>
- <https://issues.apache.org/jira/browse/SYSTEMDS-3182>

#3178: Tuple Deduplication

Taken by a bachelor student, additional team members (bachelor students) possible



▪ 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>

Additional Compression Formats for Matrix Column Groups **NEW**



- **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
- **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.
- **Huffman Encoding (in Java)**
 - Dense dictionary encoding (DDC) is an encoding scheme that utilizes a dense 'map' array of indexes and a dictionary to encode a values
 - The benefit of the current DDC encoding is that each index contained in the mapping part uses a consistent number of bits to encode, but the space allocation can be improved by further specializing this mapping into Huffman coding.
 - This project is to introduce Huffman encoded DDC columns as a specialization of the mapping of values to reduce the space used.
 - The solution has to be able to encode and decode the mapping part into a Huffman encoded semantically equivalent compressed format.
- **More information & hints**
 - On request

Compression: Specialized Co-coding Algorithm **NEW**

■ Task

- 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

- On request

Exploratory Workload-aware Compression on Intermediates **NEW**



▪ **Task**

- 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**

- On request

- **Vectorized image pre-processing (in DML)**
 - SystemDS supports image pre-processing on single 2d images. To process a large number of images at once (e.g., in DNNs), each image is represented as a single row in a matrix. Some of the image operations no longer work without materializing individual rows back into images.
 - This hinders neural network implementations and could be improved by providing image processing techniques in linear algebra that allows tensor-like access on matrices.
 - The task is to implement as many of the image processing techniques on row-linearized images as time allows.
 - For concrete examples of image operations, see <https://github.com/apache/systemds/tree/main/scripts/builtin> (prefix “img”); for instance, `img_brightness.dml` supports 2d and linearized images, while `img_cutout.dml` does not support linearized images yet, so the respective built-in functions need to be added.
- **Push-down of image pre-processing to BLAS/MKL (in Java and C/C++)**
 - OpenBLAS and IntelMKL/OneAPI all contain image pre-processing kernels that ease the development of image processing pipelines.
 - This project is to implement internal calls from SystemDS to image processing kernels via native calls in Java.
- **More information & hints**
 - On request

Convert Neural Network Layers to Built-in Functions **NEW**

▪ **Task** (in Java and DML)

- This project is to convert SystemDS's existing implementations of Neural Network (NN) layers into built-in functions while also adding functional tests for the individual layers
- Previously, all NN building blocks were using the source keyword in SystemDS and therefore are harder to properly integrate into our compiler and optimizer
- To unify the access similarly to our other builtin algorithms and functions this task is to move the NN layers into built-in functions with naming scheme x_fw and x_bw and modifying their interfaces to use lists as arguments for standard components
- Furthermore, each moved primitive has to get new dedicated tests

▪ **More information & hints**

- On request

Optimize I/O Path of SystemDS Python Interface **NEW**



- **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
- **More Information & Hints**
 - On request

- **Task** (in Java and DML)
 - Currently SystemDS relies on the CommonMath library for some primitives: eigen, cholesky decomposition, decomposition solver, SVD, LUDecpmposition, QRdecomposition (solve), which comes with three drawbacks
 - 1. We have to transfer our allocation into ones understood by CommonMath
 - 2. The CommonMath implementations is single threaded
 - 3. The CommonMath implementation does not allow the Spark backend to be utilized
 - This project is to choose and implement alternative executions of one or more of these mathematical operations that return the same values, but that are
 - 1. parallelized (local and/or distributed)
 - 2. vectorized
 - 3. implemented such that they can be executed with the Spark backend (optional)
 - As a part of the project, the individual operations have to be considered if they can be implemented directly in DML with primitives which would be preferred, otherwise internal numerically stable solutions are an option
- **More information & hints**
 - On request

Extended Performance Test Suite **NEW**

- **Task** (preferably Java/Python, other languages on request)
 - SystemDS comes with its own performance test suite. The content of this suite is run at various intervals to detect regressions in the implementation of the internals of the system.
 - This task is to execute the suite on various git commits and releases (From SystemML 1.2 time and onwards) to then summarize the results automatically and construct plots and reports of system performance over time.
 - Furthermore the implementation should be easy to use for subsequent performance suite runs results, while preferably not modifying the performance suite.
- **More information & hints**
 - On request

Hidden Markov Model Built-in Function **NEW**

- **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.
- **More information & hints**
 - On request

Heuristic-based transformation spec generator **NEW**



▪ **Task**

- This project is to define a new method that heuristically generate transformcode 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**

- On request

■ 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>
- Further details on request

■ 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

- On request

- **Alternative Linear Algebra Kernels**
 - E.g., transposition-aware matrix multiplications
 - E.g., Intel OneAPI instead of Intel MKL
 - Implementation in Java and C/C++
- **Extended Matrix Multiplication Chain Optimization**
 - Implementation in Java
- **Auto Differentiation**
 - Implementation in Java and DML
- **Loop Vectorization**
 - Implementation in Java
- **Extended Common Subexpression Elimination**
 - Implementation in Java
- **Extended I/O Framework: Readers/Writers for More File Formats (NetCDF, HDF5, Arrow)**
 - Implementation in Java
- **More Sparse Representations NEW (e.g., Double Compressed CSR)**
 - Implementation in Java
- **More information & hints**
 - On request

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