List of Project Topics (Proposals)

Last update: Apr 19, 2024

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 mainly in C++

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

Topics in DAPHNE

Implementation mainly in C++

#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 and intermediate representation (e.g., new operations for relational algebra)
    - 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, word embedding)
    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 representation of string matrices, kernels for string processing, and I/O for string-valued files)
- **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 the support is still a proof-of-concept
- **Task** (in C++)
  - Enable end-to-end sparsity exploitation in DAPHNE.
  - This includes state-of-the-art sparsity estimation, automatic selection of dense/sparse representation, kernels for processing sparse matrices efficiently, and the integration into DAPHNE's vectorized engine
  - Goal: Significant performance improvement on well-selected data sets and ML or graph algorithms expressed in DaphneDSL
- **More information & hints**
  - https://github.com/daphne-eu/daphne/issues/500
  - Contact: Patrick Damme

#691 Hybrid Compilation

- **Motivation**
  - By default, DAPHNE lowers domain-specific operations from linear and relational algebra operations to calls to pre-compiled C++ kernels (physical operators).
  - However, being based on MLIR/LLVM, on-the-fly code generation for these operations is an alternative option.
  - Interestingly, both approaches have their pros and cons, so a hybrid approach (choosing pre-compiled kernels for some operations and using code generation for others) is desirable.
- **Task** (in C++)
  - Extend DAPHNE's existing proof-of-concept for a hybrid compilation chain by additional features such as:
    - Support for the Frame data type and relational algebra operations thereon
    - Support for sparse matrix data types (e.g., CSR) and linear algebra operations thereon
    - Support for zero-copy views into row/column segments of matrices and frames
    - Integration of code-generated operations with DAPHNE's vectorized pipelines
    - Automatic decision on whether to use a pre-compiled kernel or to generate code
- **More information & hints**
  - https://github.com/daphne-eu/daphne/issues/691
  - Contact: Philipp Ortner

#693 Codegen-only Compilation Pipeline for LA Operations

- **Motivation**
  - By default, DAPHNE lowers domain-specific operations from linear and relational algebra operations to calls to pre-compiled C++ kernels (physical operators).
  - However, being based on MLIR/LLVM, on-the-fly code generation for these operations is an alternative option, which could, e.g., simplify targeting hardware accelerators through existing MLIR/LLVM backends.
- **Task** (in C++)
  - Implement a proof-of-concept for a compilation pipeline in DAPHNE, which only involves code-generated operations, thereby reducing the compatibility challenges of a hybrid compilation chain.
  - For simplicity, you can focus on linear algebra operations, supporting dense and sparse matrices.
  - The resulting code may target x86 CPUs or, optionally CUDA-enabled GPUs.
- **More information & hints**
  - Working on the GPU aspect is optional and requires a GPU with CUDA support
  - https://github.com/daphne-eu/daphne/issues/693
  - Contact: Philipp Ortner

#696 Cost-aware Function Specialization

- **Motivation**
  - DaphneDSL supports user-defined functions with typed and untyped parameters and results.
  - DAPHNE's MLIR-based optimizing compiler automatically infers the types and properties (e.g., shape, sparsity) of intermediate results in a DaphneDSL script and uses this information for various optimizations.
  - Function calls pose a special challenge as a function must be generic, but should also benefit from information on the inputs to further optimize the code.
  - To fully exploit this information inside functions, DAPHNE currently creates a separate copy of each function per call-site and specializes it with the information available on the parameters, which can be quite costly.
- **Task** (in C++)
  - Implement a pass in DAPHNE's MLIR-based optimizer, which automatically decides in which cases it makes sense to specialize a function, such that a balance between code size and improved runtime performance is achieved.
- **More information & hints**
  - https://github.com/daphne-eu/daphne/issues/696
  - Contact: Patrick Damme

#690 IDE/Tooling Support for DaphneDSL and DaphneIR

- **Motivation**
  - DaphneDSL is DAPHNE's domain-specific language for integrated data analysis pipelines.
  - Its syntax is inspired by languages like Python, R, and C.
  - DaphneDSL can be written in any text editor, but support in an integrated development environment would increase user productivity.
- **Task**
  - Implement support for DaphneDSL in a widely-used IDE (preferably VS Code), including a LSP and TreeSitter.
  - The tool should be connected to the DAPHNE compiler, especially to its features for type/shape/property inference in order to augment the DaphneDSL code with additional information
- **More information & hints**
  - https://github.com/daphne-eu/daphne/issues/690
  - Contact: Philipp Ortner

Topics in Apache SystemDS

Implementation mainly in Java and DML (SystemDS's R-like domain-specific language)

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

- **Task** (in DML)
  - Implement additional optimizers, e.g.
    - 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
  - Conduct an extensive experimental evaluation
- **More information & hints**
  - https://issues.apache.org/jira/browse/SYSTEMDS-3553
  - https://issues.apache.org/jira/browse/SYSTEMDS-3259
  - 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

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

Relational Operators on Top of DML

- **Task** (in Java and DML)
  - Implement typical relational algebra operators (e.g., selection, projection, join, grouping, and aggregation) based on the matrix operations from linear algebra offered by SystemDS's domain-specific language DML
  - Take inspiration from the paper: He et al.: Query Processing on Tensor Computation Runtimes (PVLDB, 2022)
- **More information & hints**
  - Contact: Matthias Boehm

#3694 Sequence Primitive for Neural Network Layers

- **Task** (in Java, Python, DML)
  - Construct a sequence primitive for SystemDS's Python interface, which is able to combine multiple neural network layers and perform forward and backward passes
  - The design should be inspired by Keras and PyTorch
  - Besides that, add additional neural network primitives to SystemDS's Python API
- **More information & hints**
  - https://issues.apache.org/jira/browse/SYSTEMDS-3694
  - 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

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

#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)
- **#3650 I/O Support for Cloud-optimized GeoTIFF (COG)** (in Java)
- **Various Model Debugging or Data Preprocessing Strategies** (in DML)

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

More Topics in Apache SystemDS

- **Additional State-of-the-Art Dimensionality Reduction Techniques** (in DML and Java)
  - E.g., UMAP (https://arxiv.org/abs/1802.03426)
  - Including nearest neighbor search and cost function optimization
- **Embed-Search-Align** (in DML and Java)
  - Approach described in https://arxiv.org/abs/2309.11087
  - On top of SystemDS and as a Stand-alone Baseline
  - Including self-supervised learning, transformer model, and vector database
- **More information & hints**
  - Contact: Ramon Schöndorf

Stand-alone and Cross-cutting Topics

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

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

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 or other open-source systems possible, but contributions might be more difficult to get accepted
  - **If you would like to propose your own topic**, **approach me by email by Apr 23, 23:59 CEST**;
    in any case, **also fill in the poll regarding the topic selection with your preferred topics from the list above**