DAPHNE

DAPHNE: Integrated **D**ata **A**nalysis **P**ipelines for Large-Scale Data Management, **H**PC, and Machi**ne** Learning

Patrick Damme
TU Graz & Know-Center GmbH

Oral communication @ ISPDC 2022, Basel, Switzerland, July 12, 2022

This project has received funding from the European Union's Horizon 2020 research and innovation programme under agreement number 957407.

https://daphne-eu.eu/

Modern Data-driven Applications

DAPHNE

Biomedical Engineering

ML-assisted Manufacturing

Natural Sciences

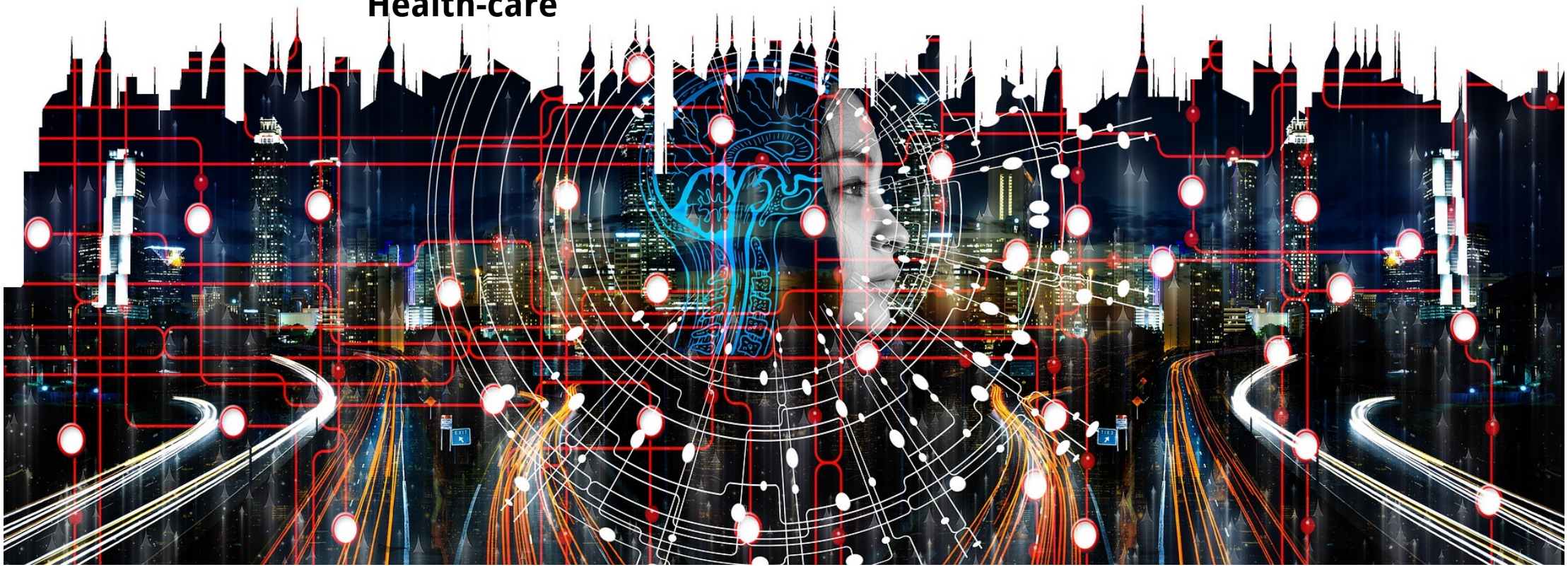
Transportation

Finance

Remote Sensing

+ many more

Health-care



Integrated Data Analysis (IDA) Pipelines

DAPHNE

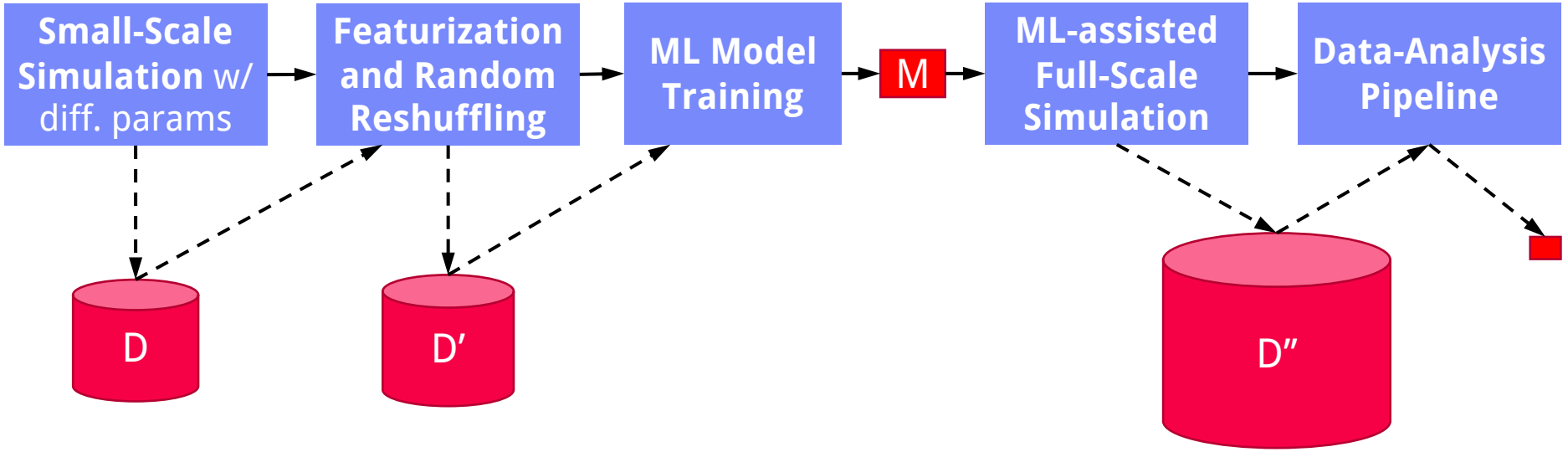
DM + **ML** + **HPC**

Data Management & query processing

Machine Learning training & scoring

High-Perf. Computing custom codes & simulations

Example: ML-assisted simulation



Challenges

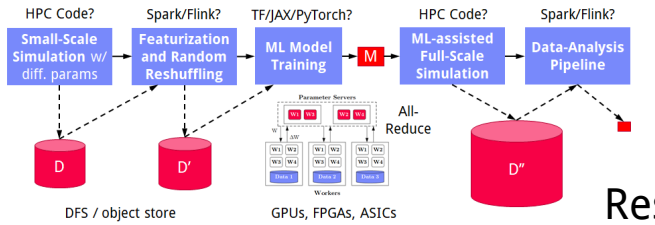
DAPHNE

- **Deployment Challenges**

Dev Teams

Programming Models

Different Systems/ Libraries



Cluster Under-utilization

Data/File Exchange

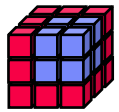
Resource Managers

➔ DAPHNE Overall Objective: Open and extensible system infrastructure

- **Hardware Challenges**
  - DM+ML+HPC share compilation and runtime techniques / converging cluster hardware
  - **End of Dennard scaling:** P = α CFV2 (power density 1)
  - **End of Moore's law**
  - **Amdahl's law:** sp = 1/s

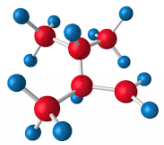
  ➔ Increasing Specialization

#1 Data Representations

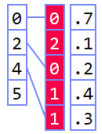


dense

graph



compressed



sparse

Sparsity Exploitation from Algorithms to HW

#2 Data Placement

Local vs **distributed**

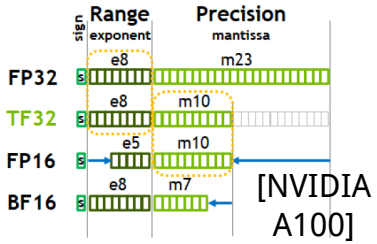
CPUs/ NUMA

GPUs

FPGAs/ ASICs

#3 Data (Value) Types

FP32, FP64, INT8, INT32, INT64, UINT8, BF16, TF32, FlexPoint



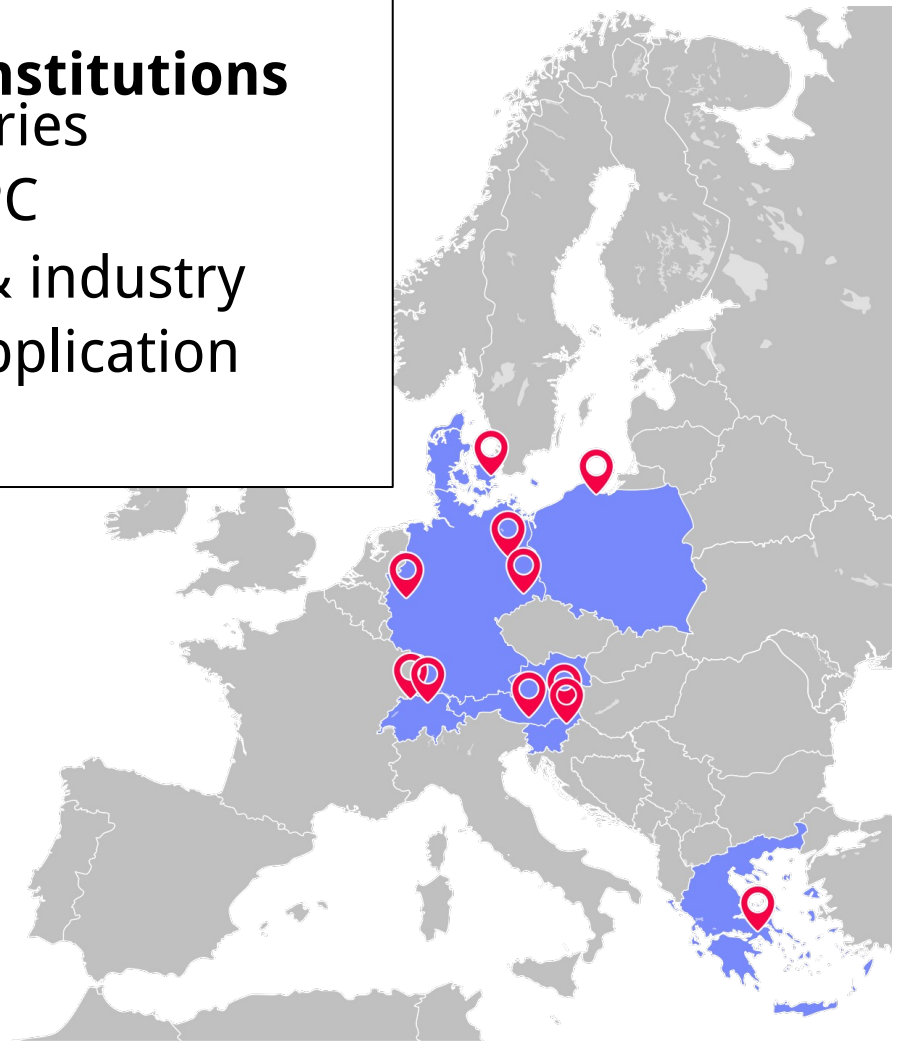
[NVIDIA A100]

Project Consortium

DAPHNE

13 partner institutions
from 7 countries

- DM, ML, HPC
- Academia & industry
- Different application domains



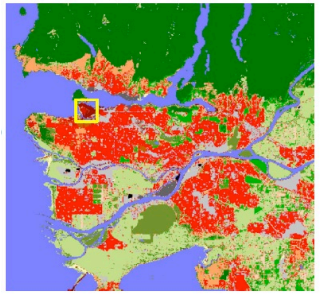
- Know-Center GmbH (**coordinator**), Austria
- AVL List GmbH, Austria
- Deutsches Zentrum fuer Luft- und Raumfahrt e.V., Germany
- Eidgenoessische Technische Hochschule Zuerich, Switzerland
- Hasso-Plattner-Institut for Digital Engineering gGmbH, Germany
- Institute of Communication and Computer Systems, Greece
- Infineon Technologies Austria AG, Austria
- , Intel Technology Poland sp. z o.o., Poland
- IT-Universitetet i København, Denmark
- Kompetenzzentrum Automobil- und Industrieelektronik GmbH, Austria
- Technische Universität Dresden, Germany
- Univerza v Mariboru, Slovenia
- Universitaet Basel, Switzerland

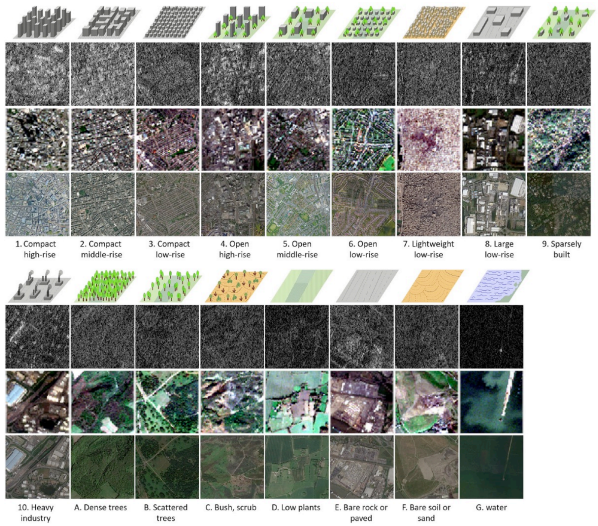
Example Use Cases

[Xiao Xiang Zhu et al: So2Sat LCZ42: A Benchmark Dataset for the Classification of Global Local Climate Zones. **GRSM 8(3) 2020**]

[So2Sat LC42: https://mediatum.ub.tum.de/1454690]

- **DLR Earth Observation**
  - **ESA Sentinel-1/2** datasets → 4PB/year
  - Training of local climate zone classifiers on **So2Sat LCZ42** (15 experts, 400K instances, 10 labels each, ~55GB HDF5)
  - **ML pipeline:** preprocessing, ResNet-20, climate models





- **IFAT Semiconductor Ion Beam Tuning**
- **KAI Semiconductor Material Degradation**
- **AVL Vehicle Development Process** (ejector geometries, KPIs)

- **ML-assisted simulations, data cleaning, augmentation**
- **Cleaning during exploratory query processing**

System Architecture

DaphneLib (API) — Python API w/ lazy evaluation

DaphneDSL (Domain-specific Language)

MLIR

MLIR-Based Compilation Chain

DaphneIR (MLIR Dialect)

Optimization Passes

New Runtime Abstractions for Data, Devices, Operations

Hierarchical Scheduling

Device Kernels (CPU, GPU, FPGA, Storage)

Vectorized Execution Engine (Fused Op Pipelines)

Sync/Async I/O Buffer/Memory Management

Local (embedded) and Distributed Environments (standalone, HPC, data lake, cloud, DB)

Extensible Infrastructure

MLIR Dialects, Extension Catalog (new data types, kernels, scheduling algs)

Multi-level Compilation/ Runtime

Sideways Entry, DSL-level constraints (data formats & data/op placement)

Fine-grained Fusion and Parallelism

Integration w/ Resource Mgmt & Prog. Models

Language Abstractions

- **Design Principles**
  - **Frame and Matrix Operations** (coarse-grained)
  - **Data Independence** (abstract data types)
  - **Extensibility** (data types, operations, HW)

- **DSL Operations**
  - **Basic built-in** operations (RA, LA)
  - **High-level built-in** operations (e.g., SQL, PS, map on frames/matrices)
  - MLIR SCF (loops, branches)
  - **Typed and untyped functions** (hierarchy of composite primitives)
  - UDFs and external libraries

Python API DaphneLib

```
dc = DaphneContext()
G = dc.from_numpy(npG)
G = (G != 0)
c = components(G, 100, True).compute()
```

Domain-specific Language DaphneDSL

```
def components(G, maxi, verbose) {
  n = nrow(G);   // get the number of vertexes
  maxi = 100;
  c = seq(1, n); // init vertex IDs
  diff = inf;    // init diff to +Infinity
  iter = 1;
  // iterative computation of connected components
  while(diff>0 & iter<=maxi) {
    u = max(rowMaxs(G * t(c)), c); // neighbor prop
    diff = sum(u != c);    // # of changed vertexes
    c = u;                 // update assignment
    iter = iter + 1;
  }
}
```

Multiple dispatch of functions/kernels

Optimizing Compilation Chain

- **Goal:** systematic lowering from DaphneIR to kernels and LLVM
- Optimization Passes
  - **MLIR Programming Language Rewrites** (CSE, constant propagation, constant folding, branch removal, code motion/loop hoisting, function inlining / unrolling)
  - **Type and Property Inference** (e.g., types/schema, shapes/sparsity, symmetry)
  - Inter-Procedural Analysis (function specialization)
  - Algebraic Simplification Rewrites (e.g., relational/linear algebra rewrites)
  - **Operator Ordering** (e.g., join ordering/enumeration, matrix multiplication chain optimization, sum-product optimizations, data-flow-graph linearization)
  - **Generation of Fused Operator Pipelines** (selection of fused operators in DAGs, vectorization/tiling, and splitting/merging strategies of inputs/results)
  - **Memory Management** (update-in-place, reuse of allocations, garbage collection)
  - **Execution Type Selection** (local vs distributed incl. primitives caching/partitioning)
  - **Device Placement** (e.g., CPU/GPU/FPGA, multiple devices)
  - Physical Operator Selection (e.g., different join/group-by/matmult operators)

Data Representations

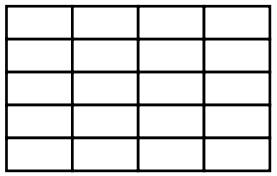
DAPHNE

- **Data Types:** Matrix, Frame, Scalar, (Tensor, List)
- **Value Types:** e.g., SI8, SI32, SI64, UI8, UI32, UI64, FP32, FP64

Local runtime

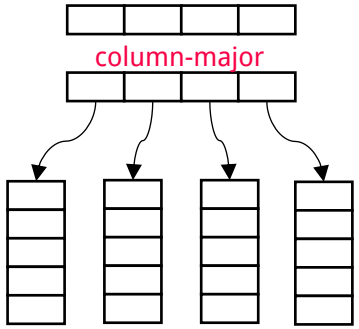
Dense Matrix

one dense array

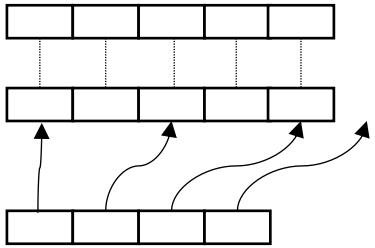


row-major

Frame



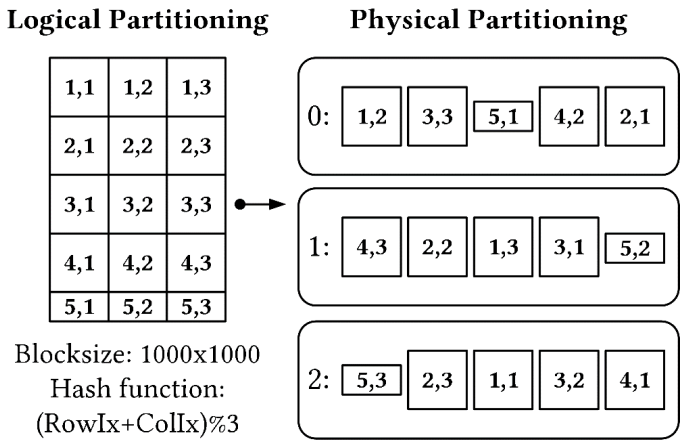
CSR Matrix



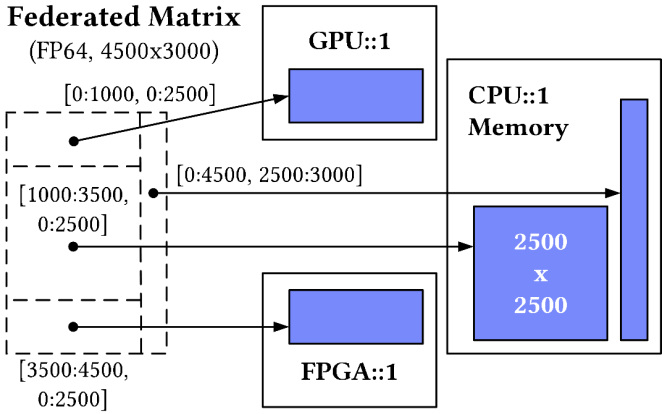
ordered rows

Distributed runtime

Distributed Collection of Tiles

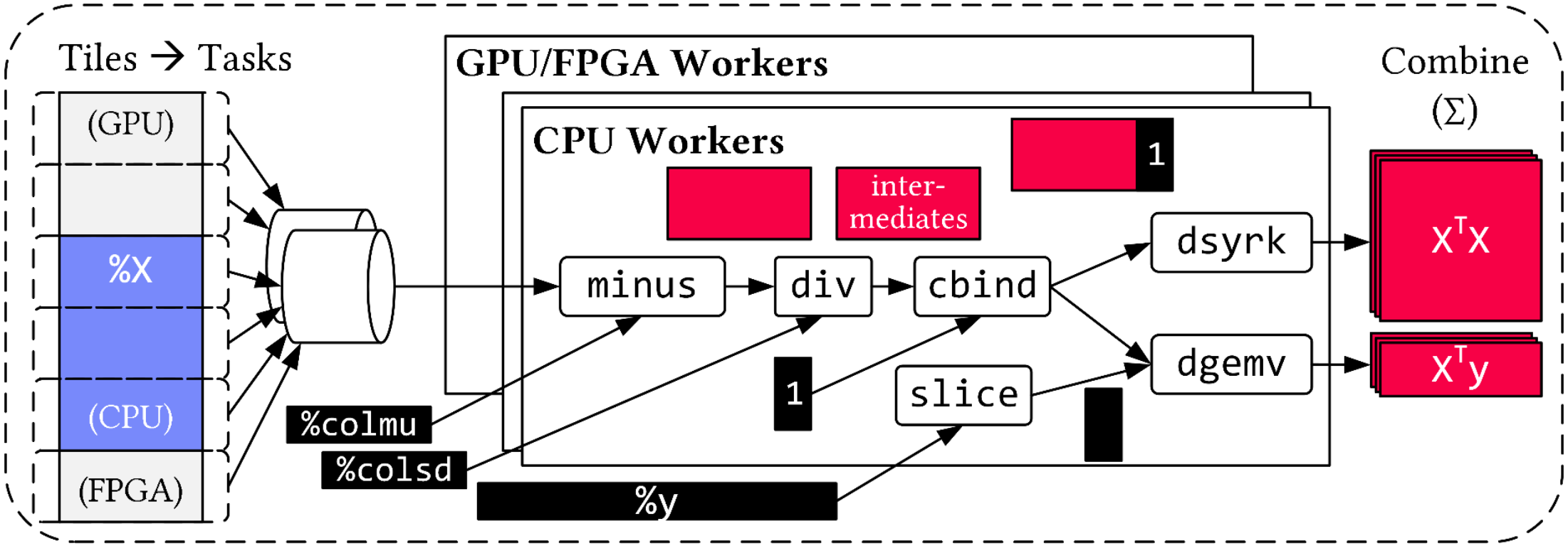


Federated Matrix/Frame



Vectorized (Tiled) Execution

```
(%9, %10) = fusedPipeline1(%X, %y, %colmu, %colsd) {
```



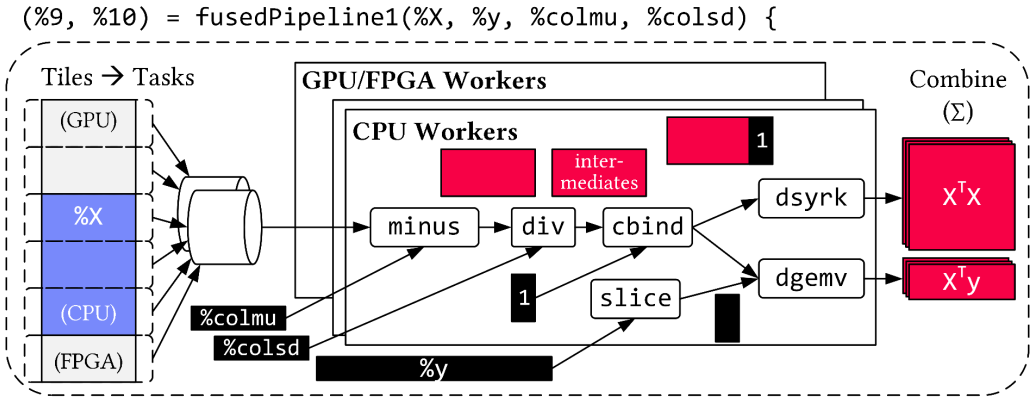
Default Parallelization Frame & Matrix Ops

Locality-aware, Multi-device Scheduling

Fused Operator Pipelines on Tiles/Scalars + Codegen

Vectorized (Tiled) Execution, cont.

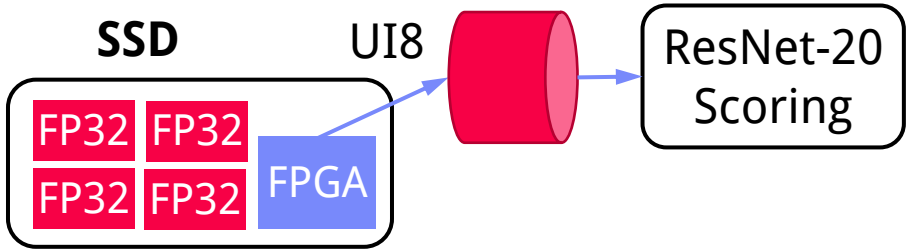
- **#1 Zero-copy Input Slicing**
  - Create view on sliced input (no-op)
  - All kernels work on views



- **#2 Sparse Intermediates**
  - Reuse dense/sparse kernels
  - Sparse pipeline intermediates for free

- **#3 Fine-grained Control**
  - Task sizes (dequeue, data access) vs data binding (cache-conscious ops)
  - Scheduling for load balance (e.g., sparse operations)

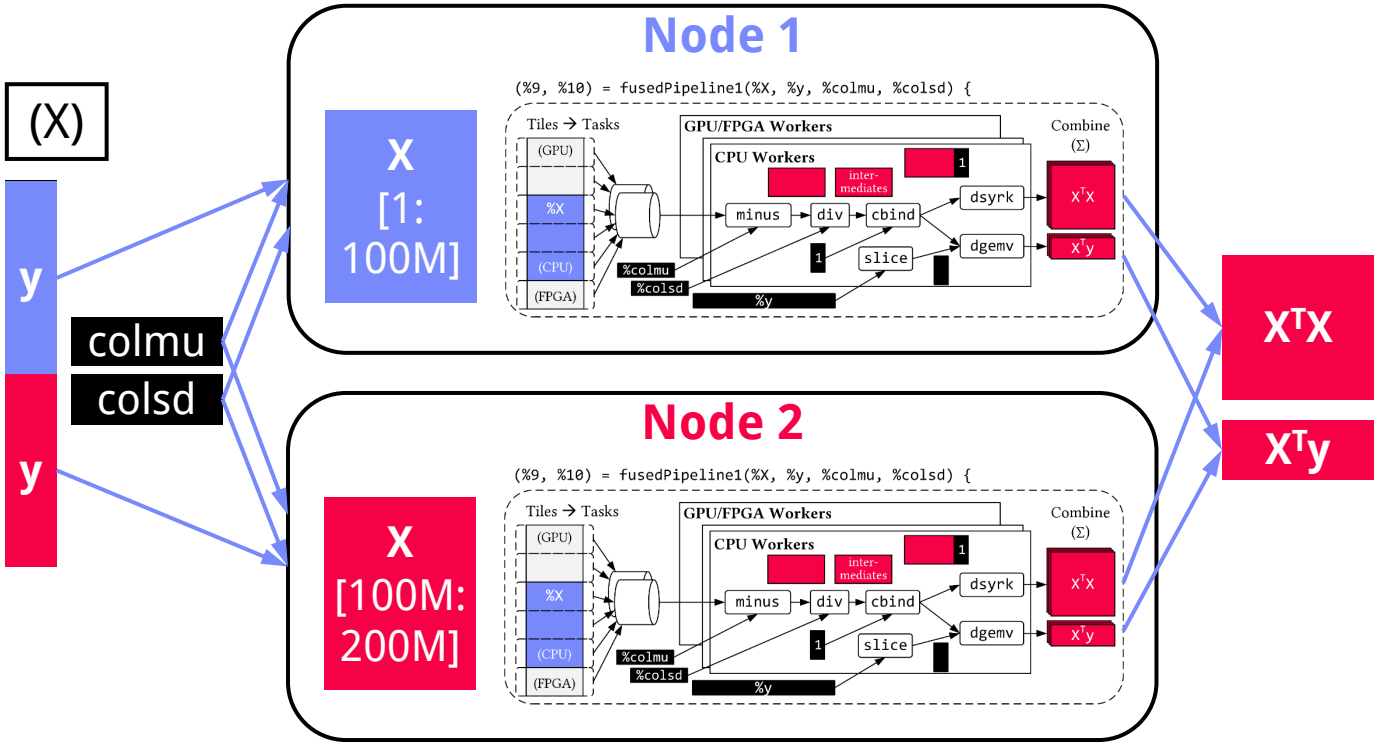
- **#4 Computational Storage**
  - Task queues connect eBPF programs, async I/O into buffers, and subsequent operator pipelines



Distributed Vectorized Execution

- Federated matrices/frames + distribution primitives
- Hierarchical vectorized pipelines and scheduling

- Coordinator
  (spawns distributed fused pipeline)
  - **#1 Prepare Inputs**
    (N/A, repartition, broadcasts, slices broadcasts as necessary)
  - **#2 Coarse-grained Tasks**
    (tasks run vectorized pipeline)
  - **#3 Combine Outputs**
    (N/A, all-reduce, rbind/cbind)



Extensibility

- **Goals for Extensibility**
  - New **data types** and **kernels** (e.g., compressed, HW devices)
  - New **optimization passes** and scheduling algorithms
  - Integration with other **MLIR dialects** (e.g., linalg)

- **#1 Extension Catalog**
  - Register kernels/data types as shared libraries
  - Type hierarchy, cost functions, constraints

Artifact	Type	Cost	Lib
compress	K-Reorg		./clib.so
mm_asic	K-Matmult		./mma.so
CompMatrix	D-Matrix		./clib.so

- **#2 DSL-level Extensibility/Configuration**
  - Data representations, data/ops placement (**constraints**)
  - **Sideways Entry:** daphnec takes DaphneDSL and DaphneIR

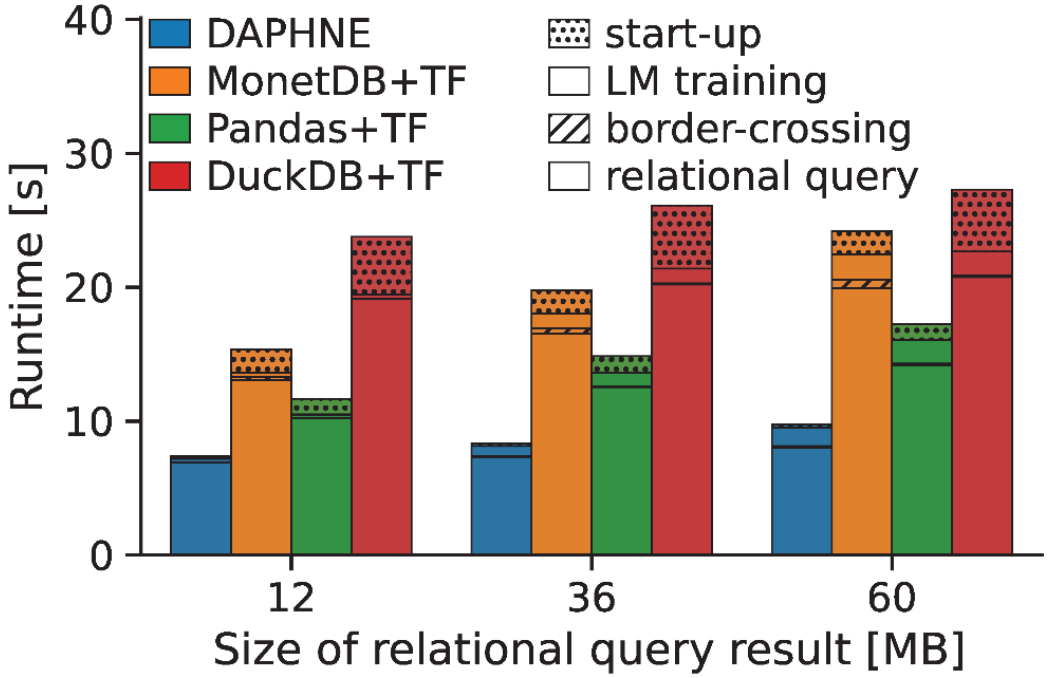
```
X = sparse(Y);
X = compress(Y);
X = device(Y, "/GPU:0");
X = Y @_gpu Z;
```

- **#3 System Internals**
  - Extended DaphneIR, new optimization passes, custom compilation chains

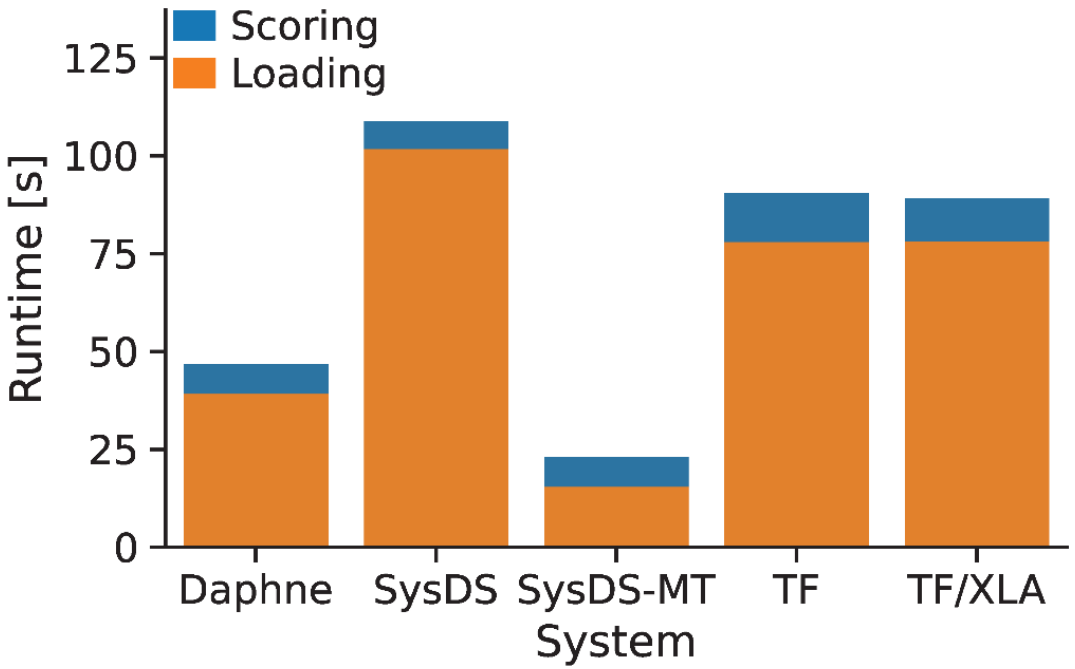
Experiments: Simple IDA Pipelines

Setup: Single node w/ 2x Intel Xeon Gold 6238 (112 vcores, 7.7 TFLOP/s), 768 GB DDR4 RAM, 12x 2TB SSDs (data), NVIDIA **T4 GPU** (8.1 TFLOP/s, 16 GB), and Intel FPGA PAC D5005 (w/ Stratix **10SX FPGA**, 32 GB) since Dec 29

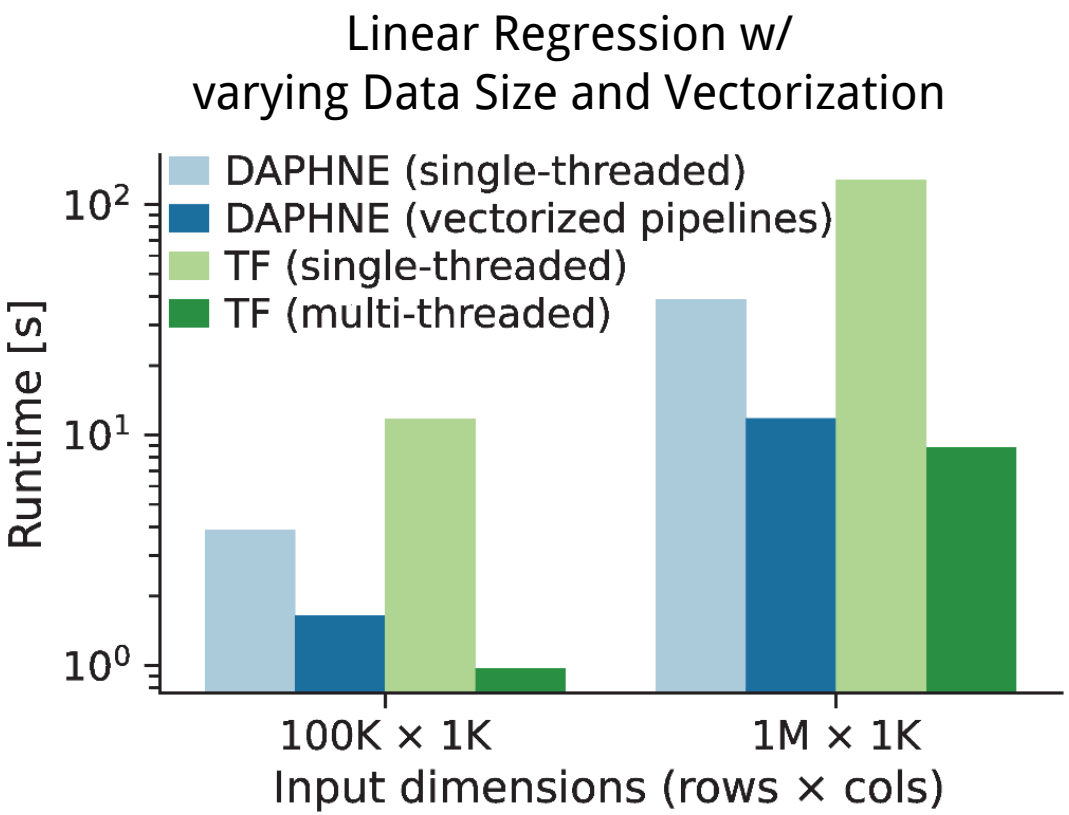
P1: TPC-H SF10 csv, query processing + linear regression training on CPUs

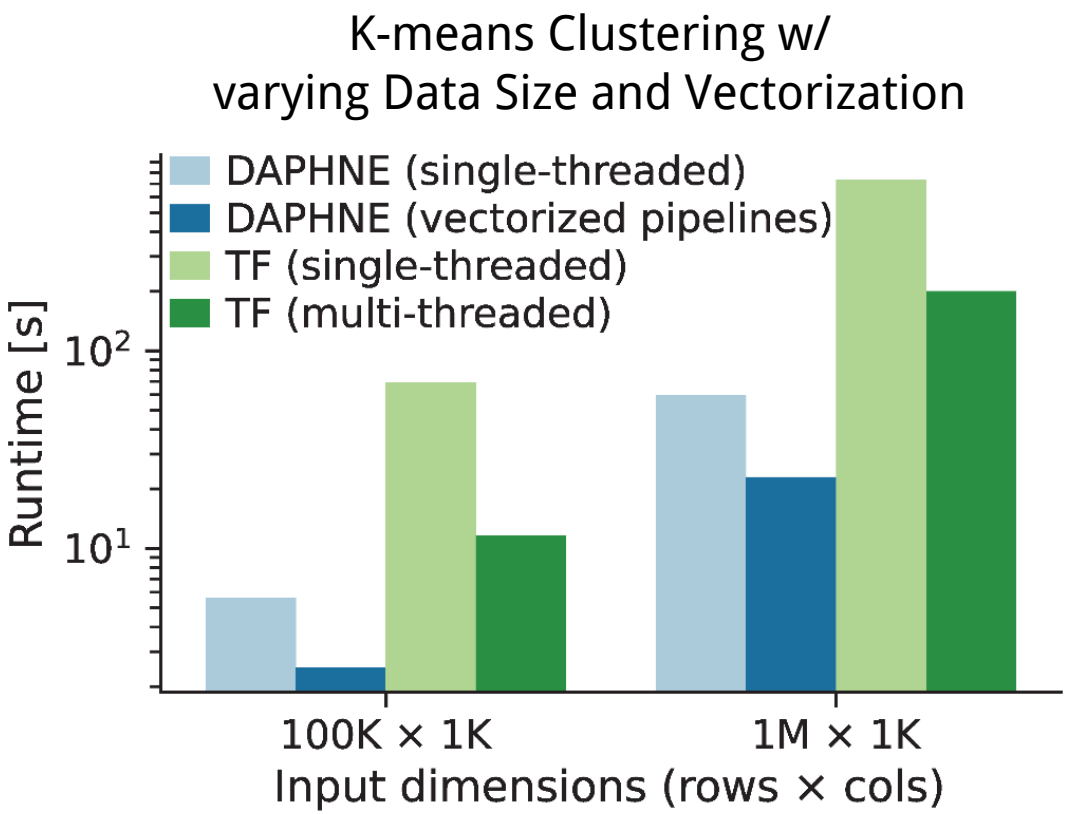


P2: So2Sat LCZ42 csv (testset), ResNet-20 scoring on GPU



Experiments: Vectorized Execution





- **Ongoing Experiments**
  - FPGA kernels on D5005, CPU+GPU vectorized pipelines
  - Distributed sparse runtime operations on Vega supercomputer
  - Sparse vectorized pipelines and scheduling algorithms

Summary

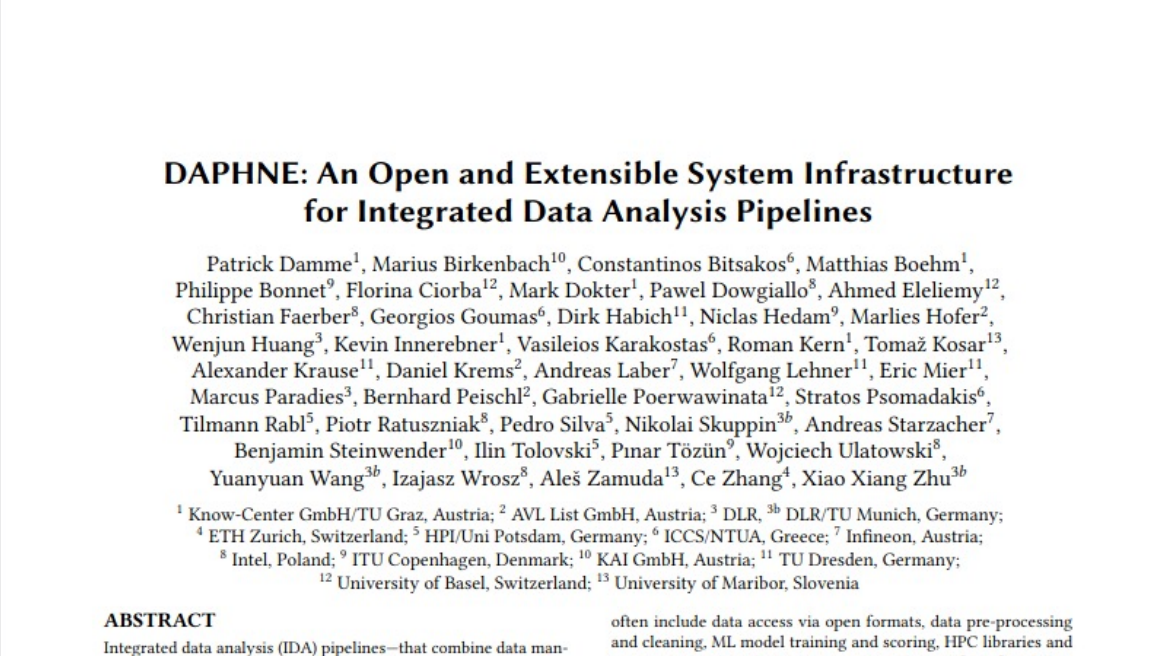
DAPHNE

DM + ML + HPC

➔ DAPHNE Overall Objective:
Open and extensible system infrastructure

- **Current Status**
  - System architecture and design
  - Initial DSL and Python API
  - Prototype of MLIR-based compiler and runtime
  - Vectorized execution (fused pipelines, scheduling)
  - GPU (and FPGA) integration, BLAS/DNN libraries, I/O primitives
  - Standalone distributed runtime w/ different distribution primitives

- **Joint Paper on System Architecture**
  - Published at CIDR 2022



DAPHNE: An Open and Extensible System Infrastructure for Integrated Data Analysis Pipelines

Patrick Damme[1], Marius Birkenbach[10], Constantinos Bitsakos[6], Matthias Boehm[1], Philippe Bonnet[9], Florina Ciorba[12], Mark Dokter[1], Pawel Dowgiallo[8], Ahmed Eleliemy[12], Christian Faerber[8], Georgios Goumas[6], Dirk Habich[11], Niclas Hedam[9], Marlies Hofer[2], Wenjun Huang[3], Kevin Innerebner[1], Vasileios Karakostas[6], Roman Kern[1], Tomaž Kosar[13], Alexander Krause[11], Daniel Krems[2], Andreas Laber[7], Wolfgang Lehner[11], Eric Mier[11], Marcus Paradies[3], Bernhard Peischl[2], Gabrielle Poerwawinata[12], Stratos Psomadakis[6], Tilmann Rabl[5], Piotr Ratuszniak[8], Pedro Silva[5], Nikolai Skuppin[3b], Andreas Starzacher[7], Benjamin Steinwender[10], Ilin Tolovski[5], Pınar Tözün[9], Wojciech Ulatowski[8], Yuanyuan Wang[3b], Izajasz Wrosz[8], Aleš Zamuda[13], Ce Zhang[4], Xiao Xiang Zhu[3b]

[1] Know-Center GmbH/TU Graz, Austria; [2] AVL List GmbH, Austria; [3] DLR, [3b] DLR/TU Munich, Germany; [4] ETH Zurich, Switzerland; [5] HPI/Uni Potsdam, Germany; [6] ICCS/NTUA, Greece; [7] Infineon, Austria; [8] Intel, Poland; [9] ITU Copenhagen, Denmark; [10] KAI GmbH, Austria; [11] TU Dresden, Germany; [12] University of Basel, Switzerland; [13] University of Maribor, Slovenia

ABSTRACT

Integrated data analysis (IDA) pipelines—that combine data man-

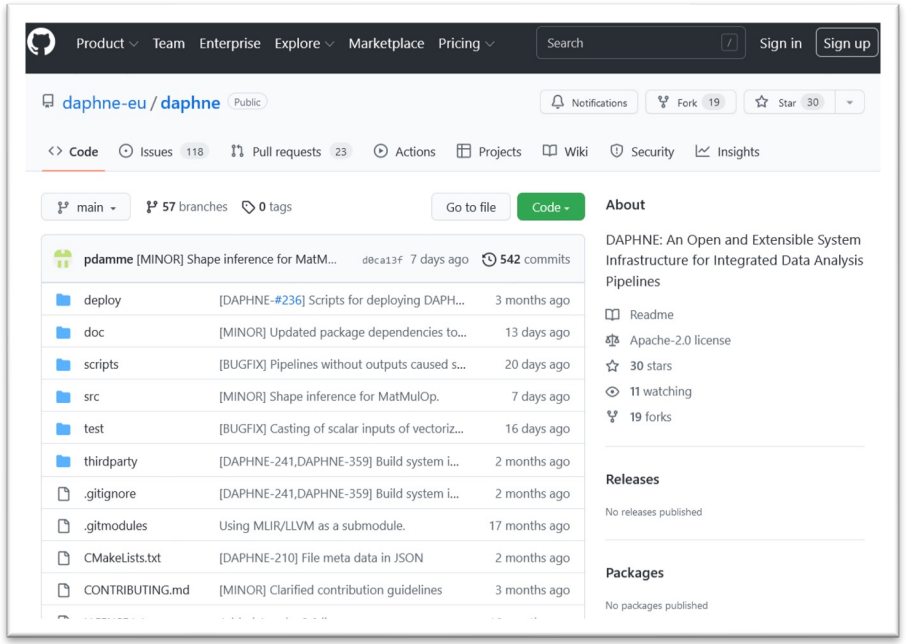
often include data access via open formats, data pre-processing and cleaning, ML model training and scoring, HPC libraries and

Further Information

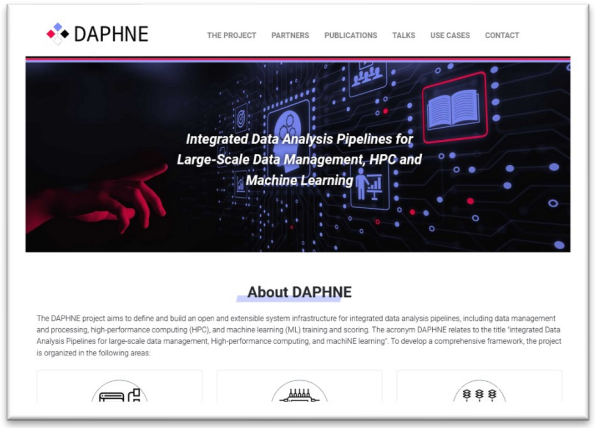
- **DAPHNE is open-source software**
  - https://github.com/daphne-eu/daphne
  - **Apache v2** license
  - Towards an inclusive dev community

➔ **Potential for collaboration in 2022-2024**

Enable researchers to experiment with new prototypes and extensions



- **Check out our website**
  - https://daphne-eu.eu
- **Follow us on twitter**
  - @daphne_eu



Backup