Integrated **D**ata **A**nalysis **P**ipelines
for Large-scale Data Management,
HPC, and Machi**ne** Learning;
DAPHNE daughter of river god Peneus
(fountains, streams), chased by Apollo

[Louvre, Paris]

DAPHNE
https://daphne-eu.eu/

The DAPHNE project is funded by the European Union's Horizon 2020 research and innovation program under grant agreement number 957407 for the time period from Dec/2020 through Nov/2024.

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

Patrick Damme, Marius Birkenbach, Constantinos Bitsakos, **Matthias Boehm**, Philippe Bonnet, Florina Ciorba, Morten Clausen, Dževad Ćoralić, Mark Dokter, Pawel Dowgiallo, Ahmed Eleliemy, Christian Faerber, Jonathan Giger, Georgios Goumas, Dirk Habich, Niclas Hedam, Marlies Hofer, Wenjun Huang, Kevin Innerebner, Vasileios Karakostas, Roman Kern, Tomaž Kosar, Thomas Krametter, Alexander Krause, Daniel Krems, Andreas Laber, Wolfgang Lehner, Eric Mier, Marcus Paradies, Bernhard Peischl, Constantin Pestka, Gabrielle Poerwawinata, Stratos Psomadakis, Tilmann Rabl, Piotr Ratuszniak, Pedro Silva, Nikolai Skuppin, Andreas Starzacher, Benjamin Steinwender, Nils Strassenburg, Ilin Tolovski, Pınar Tözün, Wojciech Ulatowski, Aristotelis Vontzalidis, Yuanyuan Wang, Izajasz Wrosz, Aleš Zamuda, Ce Zhang, Xiao Xiang Zhu

FG DB Spring Symposium 2022; March 24/25, 2022

Motivation

➔ DAPHNE Overall Objective:
Open and extensible system infrastructure

DAPHNE

- Integrated Data Analysis Pipelines
  - Open data formats, query processing
  - Data preprocessing and cleaning
  - ML model training and scoring
  - HPC, custom codes, and simulations

- **Hardware Challenges**
  - DM+ML+HPC share compilation and runtime techniques / converging cluster hardware
  - **End of Dennard scaling:** $P = \alpha\, CFV^2$ (power density 1)
  - **End of Moore's law**
  - **Amdahl's law:** sp = 1/s

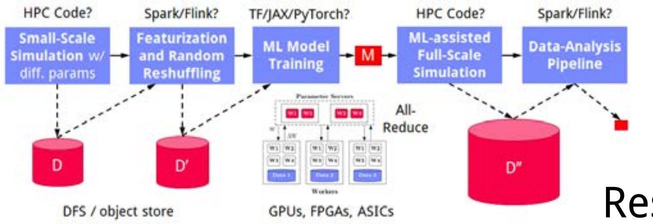
➔ Increasing Specialization

Deployment Challenges

Dev Teams

Programming Models

Different Systems/ Libraries

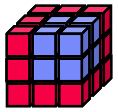


Cluster **Under-utilization**

Data/File Exchange

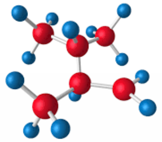
Resource Managers

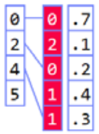
#1 Data Representations



dense

graph





sparse

compressed

Sparsity Exploitation
from Algorithms to HW

#2 Data Placement

Local vs **distributed**

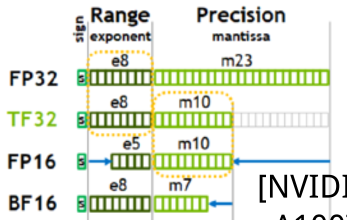
CPUs/ NUMA

GPUs

FPGAs/ ASICs

#3 Data (Value) Types

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



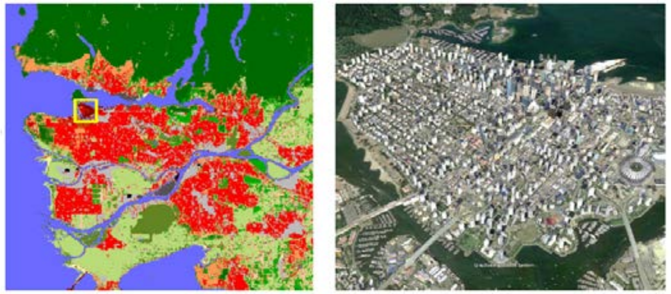
[NVIDIA A100]

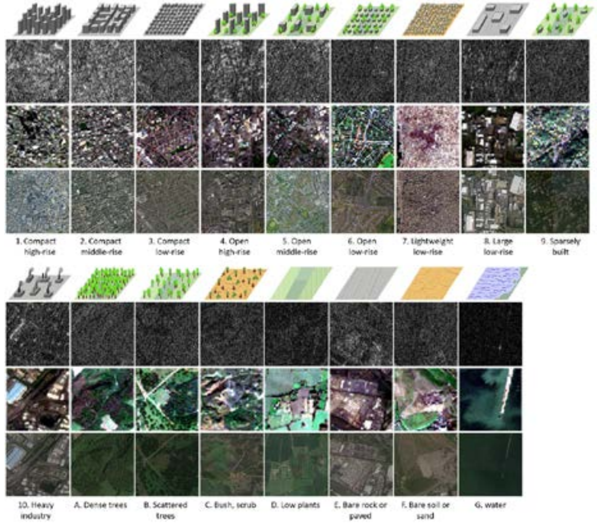
Example Use Cases

[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

[Patrick Damme et al.: **DAPHNE:** An Open and Extensible System Infrastructure for Integrated Data Analysis Pipelines, **CIDR 2022**]

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

DAPHNE

- **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

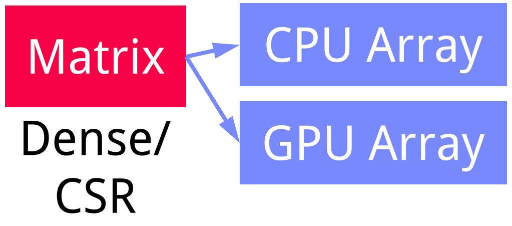
```
def components(G, maxi, verbose) {
  n = nrow(G);   // get the number of vertexes
  maxi = 100;
  c = seq(1, n); // init vertex IDs
  diff = inf;    // init diff to +Infinity
  iter = 1;
  // iterative computation of connected components
  while(diff>0 & iter<=maxi) {
    u = max(rowMaxs(G * t(c)),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

Multiple dispatch of functions/kernels

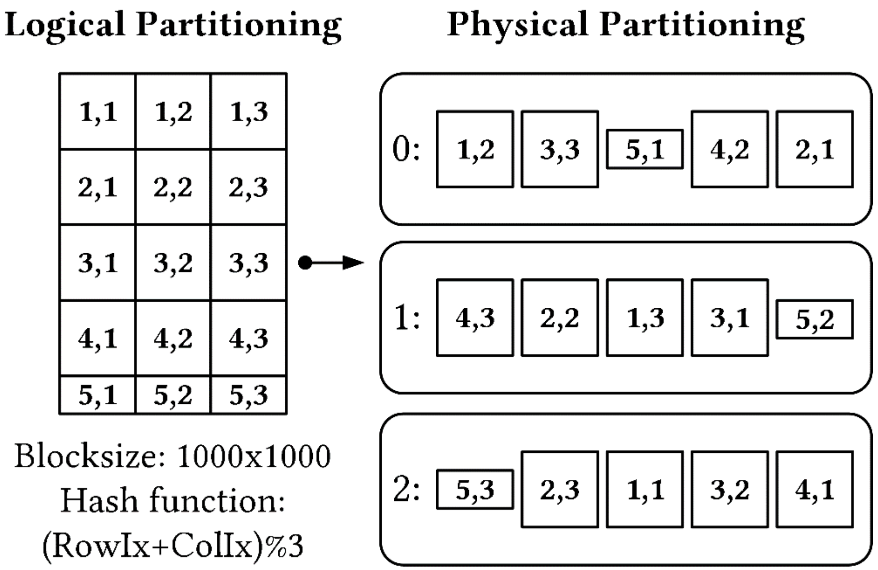
Data Representations

DAPHNE

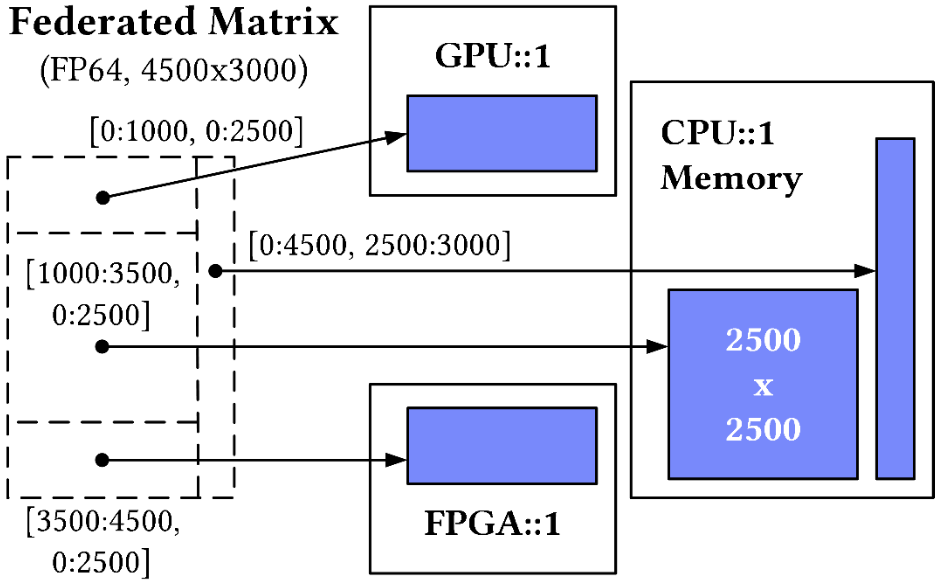
- **Data Types:** Matrix, Frame, LLVM scalars, (Tensor, List)
- **Value Types:** e.g., I8, I32, I64, UI8, UI32, UI64, FP32, FP64
- Distributed/Multi-device Data:



a) Distributed Collection of Tiles



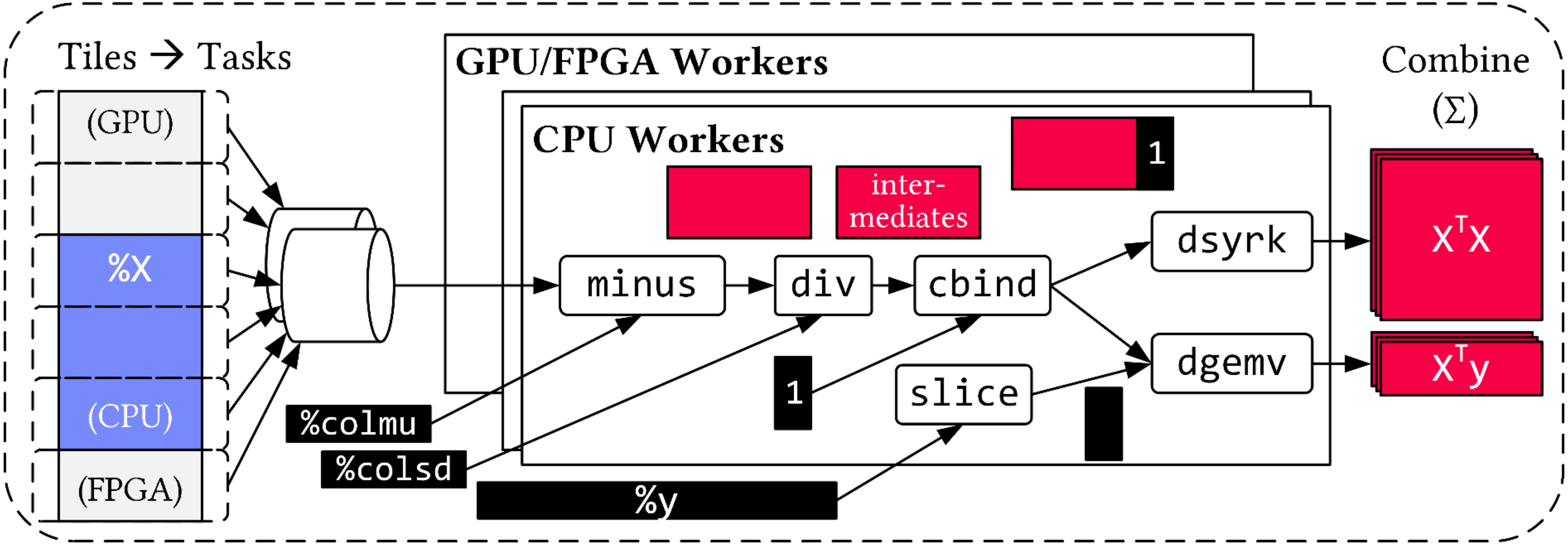
b) Federated Matrix/Frame



Vectorized (Tiled) Execution

DAPHNE

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



Default Parallelization
Frame & Matrix Ops

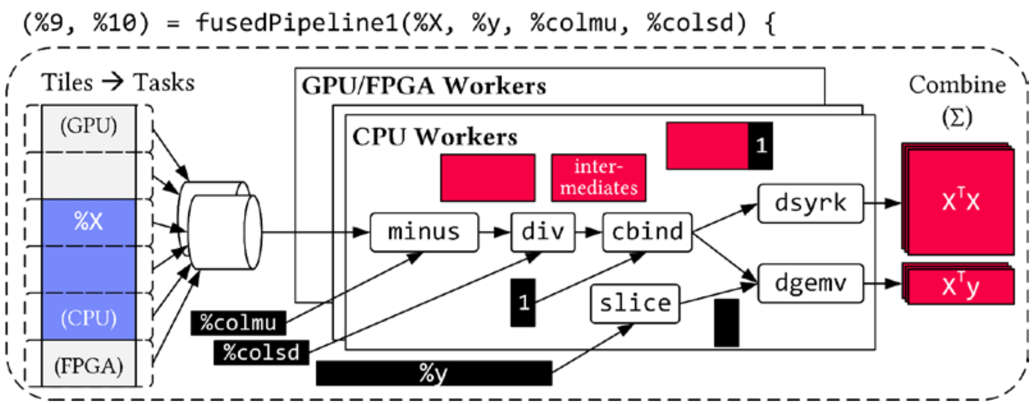
Locality-aware,
Multi-device Scheduling

Fused Operator Pipelines
on Tiles/Scalars + Codegen

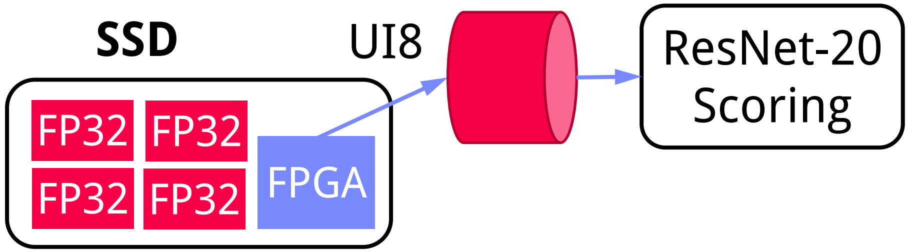
Vectorized (Tiled) Execution, cont.

DAPHNE

- **#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

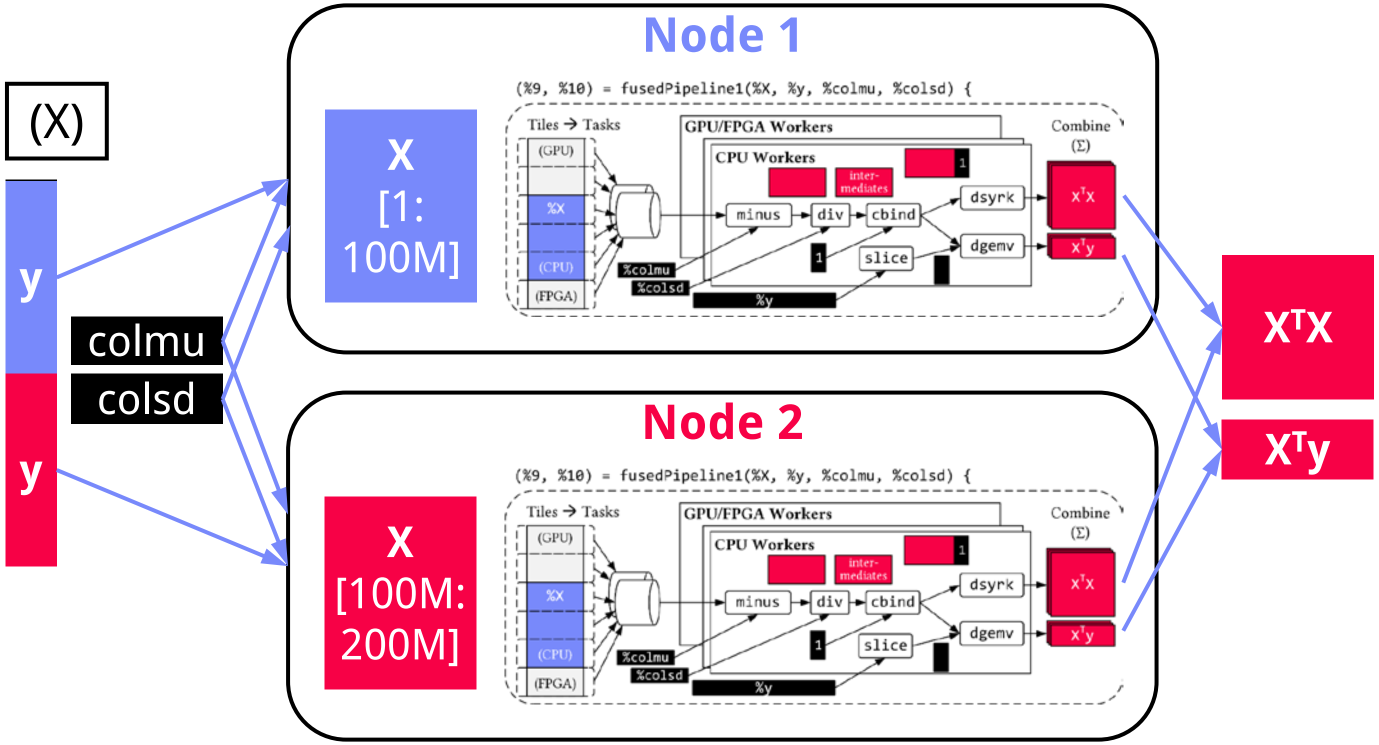


Distributed Vectorized Execution

DAPHNE

- 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)

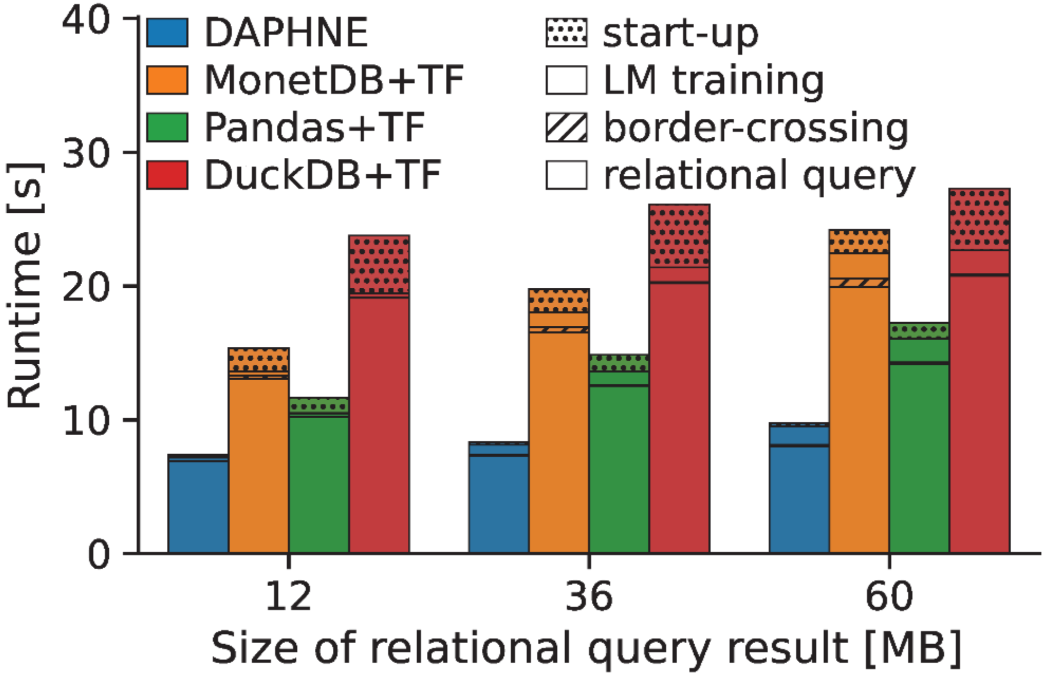


Experiments: Simple IDA Pipelines

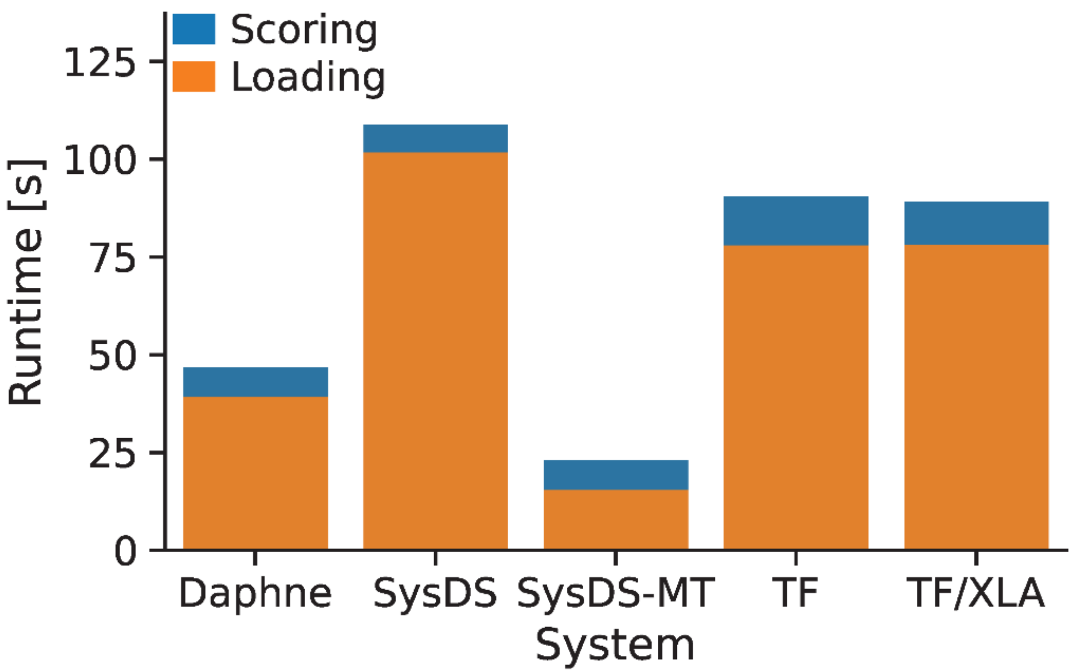
DAPHNE

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



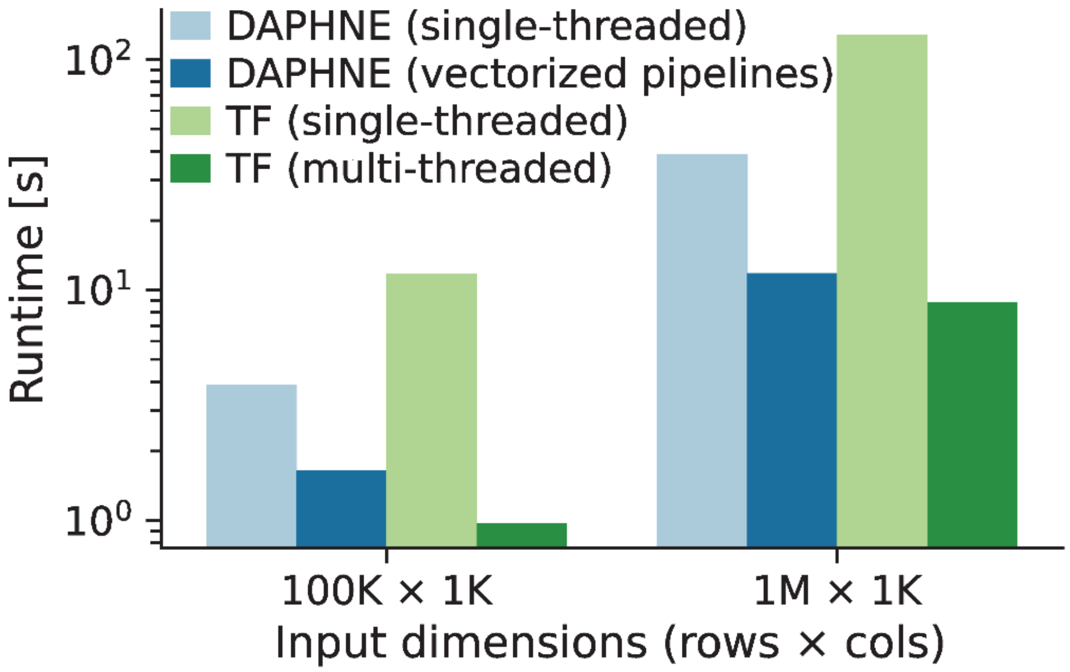
P2: So2Sat LCZ42 csv (testset), ResNet-20 scoring on GPU



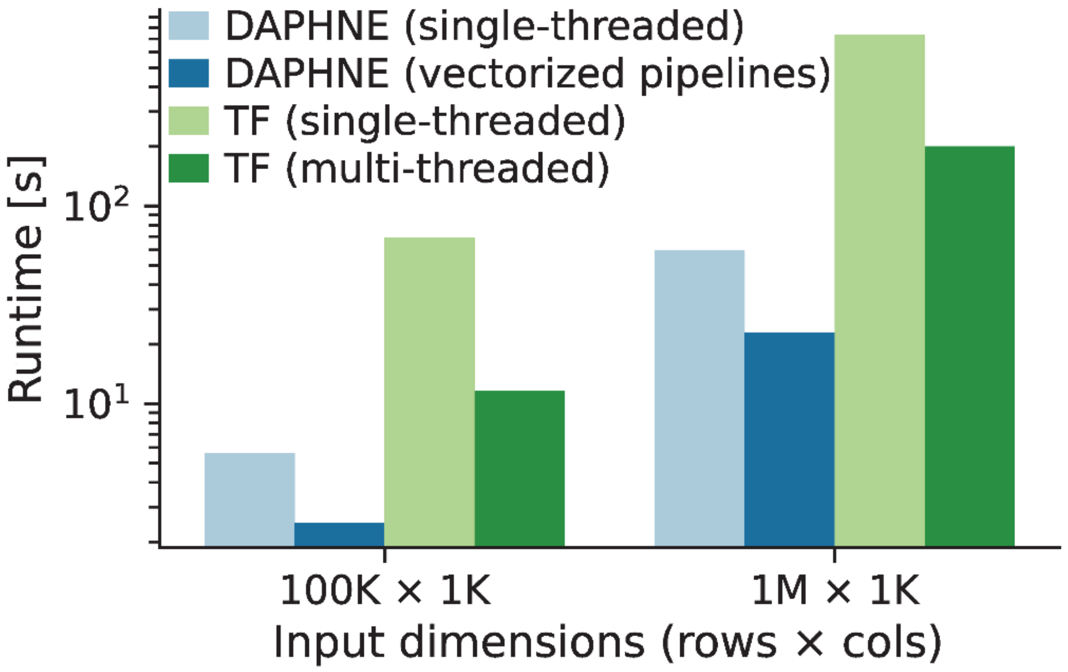
Experiments: Vectorized Execution

DAPHNE

Linear Regression w/
varying Data Size and Vectorization



K-means Clustering w/
varying Data Size and Vectorization



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

Status and Next Steps

https://daphne-eu.eu/ DAPHNE

- **Current Status**
  - System Architecture and Design
  - Initial DSL and Python API
  - MLIR-based Compiler and Runtime Prototype
  - Vectorized Execution (fused pipelines, scheduling)
  - GPU (and FPGA) Integration, BLAS/DNN Libraries
  - Standalone Distributed Runtime

➔ DAPHNE Overall Objective: Open and extensible system infrastructure

DM + HPC + ML

- **Promising Progress and Preliminary Experiments**

- **DAPHNE OSS Announcement**
  - Public **release by 03/2022**
  - **Apache v2** license
  - Towards an inclusive dev community

  ➔ Potential for collaboration in 2022-2024

Enable researchers to experiment with new prototypes and extensions