Holistic Extensibility for Integrated Data Analysis Pipelines in DAPHNE

Patrick Damme
Technische Universität Berlin

Invited Talk at CDMS@VLDB 2023, Vancouver, Canada, Aug 28, 2023

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

Modern Data-driven Applications

DAPHNE

Biomedical Engineering

ML-assisted Manufacturing

Natural Sciences

Transportation

Finance

Remote Sensing

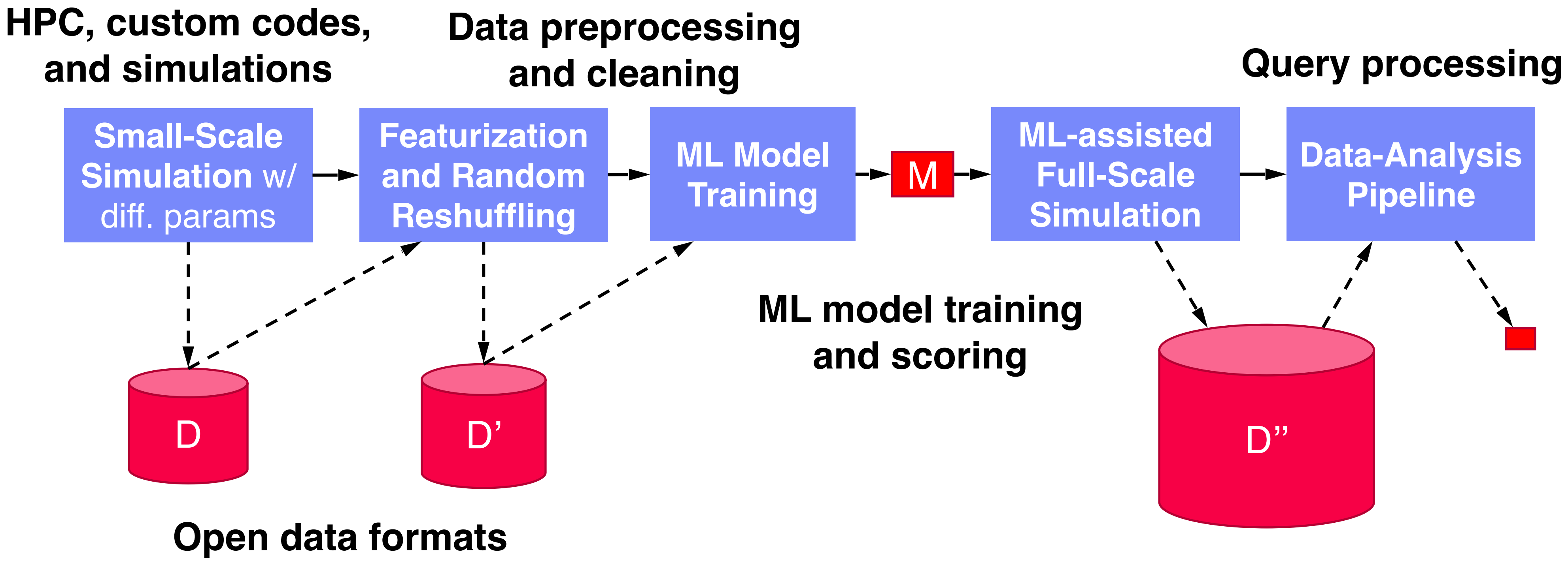
+ many more

Health-care

Integrated Data Analysis (IDA) Pipelines

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

Data Management + Machine Learning + High-Perf. Computing



Challenges

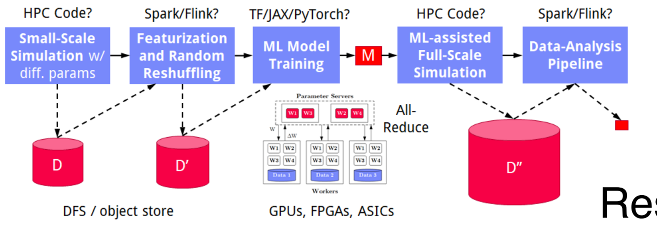
DAPHNE

- **Deployment Challenges**

Dev Teams

Programming Models

Different Systems/ Libraries



Cluster Under-utilization

Data/File Exchange

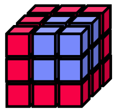
Resource Managers

DAPHNE Overall Objective:
Open and extensible system infrastructure

Increasing specialization

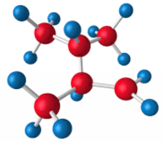
- **Hardware Challenges**
  - DM+ML+HPC share compilation and runtime techniques / converging cluster hardware
  - **End of Dennard scaling**
  - **End of Moore's law**
  - **Amdahl's law**

#1 Data Representations

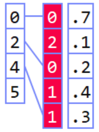


dense

graph



compressed



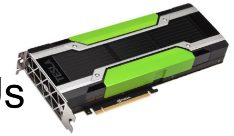
sparse

Sparsity Exploitation from Algorithms to HW

#2 Data Placement

Local vs **distributed**

CPUs/ NUMA

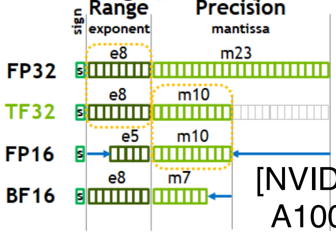


GPUs

FPGAs/ ASICs

#3 Data (Value) Types

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



[NVIDIA A100]

Project Consortium

DAPHNE

- **14 Partner Institutions**
  from 7 European Countries

- Different Backgrounds
  - Data Management
  - High-Performance Computing
  - ML Systems
  - ML/NLP/Graph Algorithms
  - Simulation & Optimization

- Different Application Domains

- Academia and Industry

Know-Center GmbH **(coordinator)**, Austria

AVL List GmbH, Austria

Deutsches Zentrum für Luft- und Raumfahrt e.V., Germany

Eidgenössische Technische Hochschule Zürich, Switzerland

Hasso-Plattner-Institut for Digital Engineering gGmbH, Germany

Institute of Communication and Computer Science, Greece

Infineon Technologies Austria AG, Austria

Intel Technology Poland sp. z o.o., Poland

IT-Universitetet i København, Denmark

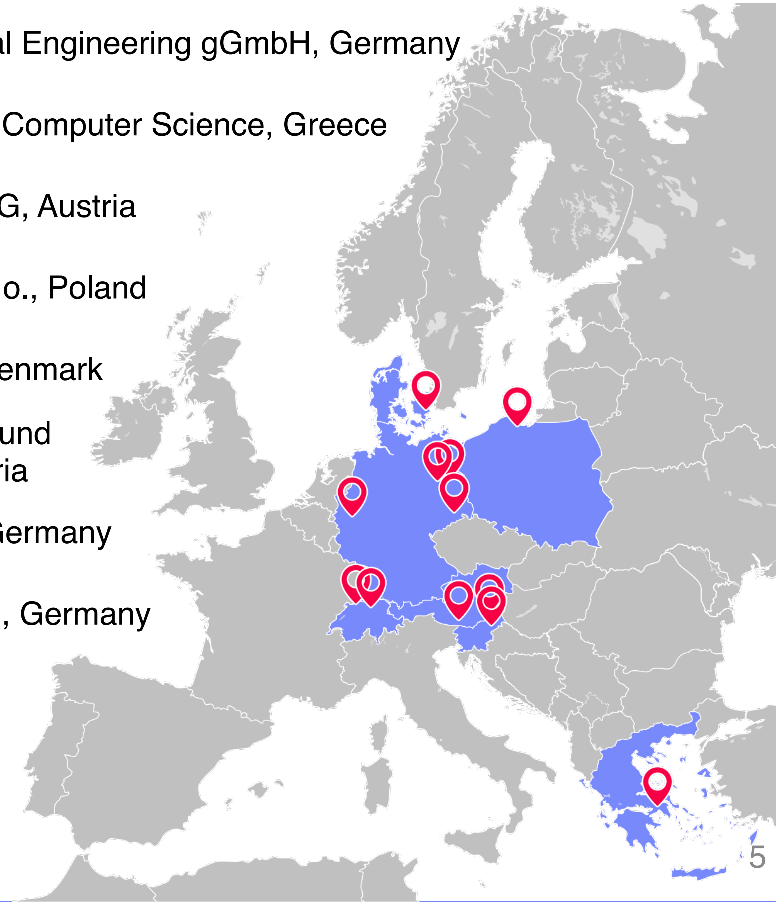
Kompetenzzentrum Automobil- und Industrieelektronik GmbH, Austria

Technische Universität Berlin, Germany

Technische Universität Dresden, Germany

Univerza v Mariboru, Slovenia

Universität Basel, Switzerland

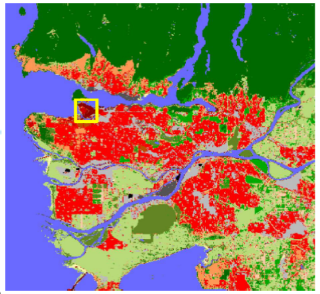


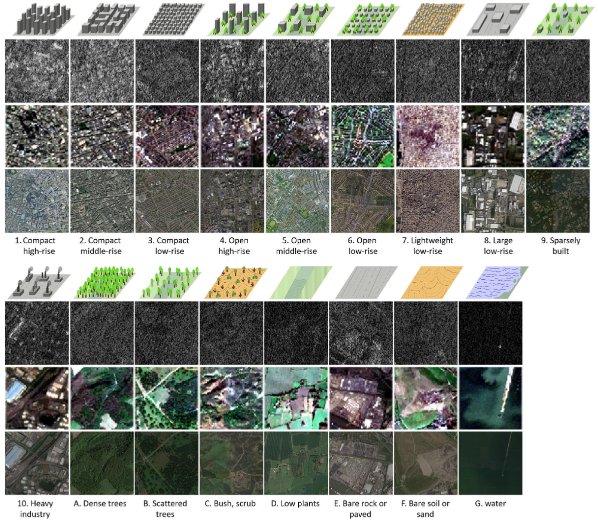
Example Use Cases

[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





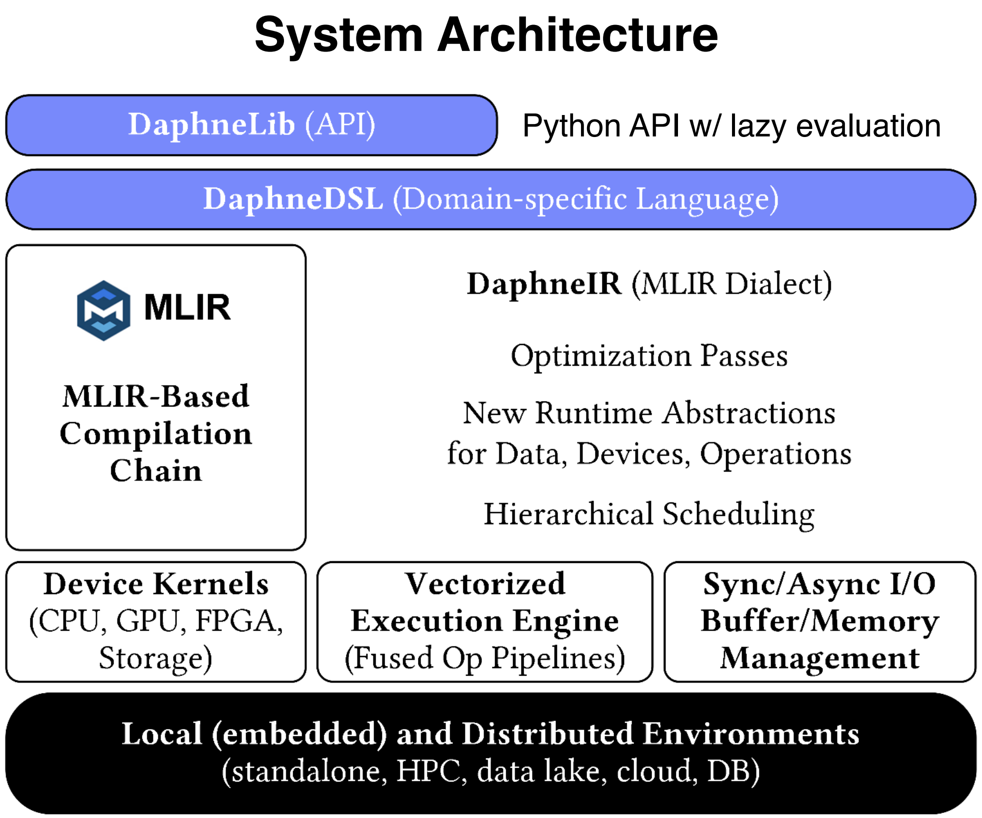
- **IFAT Semiconductor Ion Beam Tuning**
- **KAI Semiconductor Material Degradation**
- **AVL Vehicle Development Process** (ejector geometries, KPIs)

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

Overview of DAPHNE

System Architecture

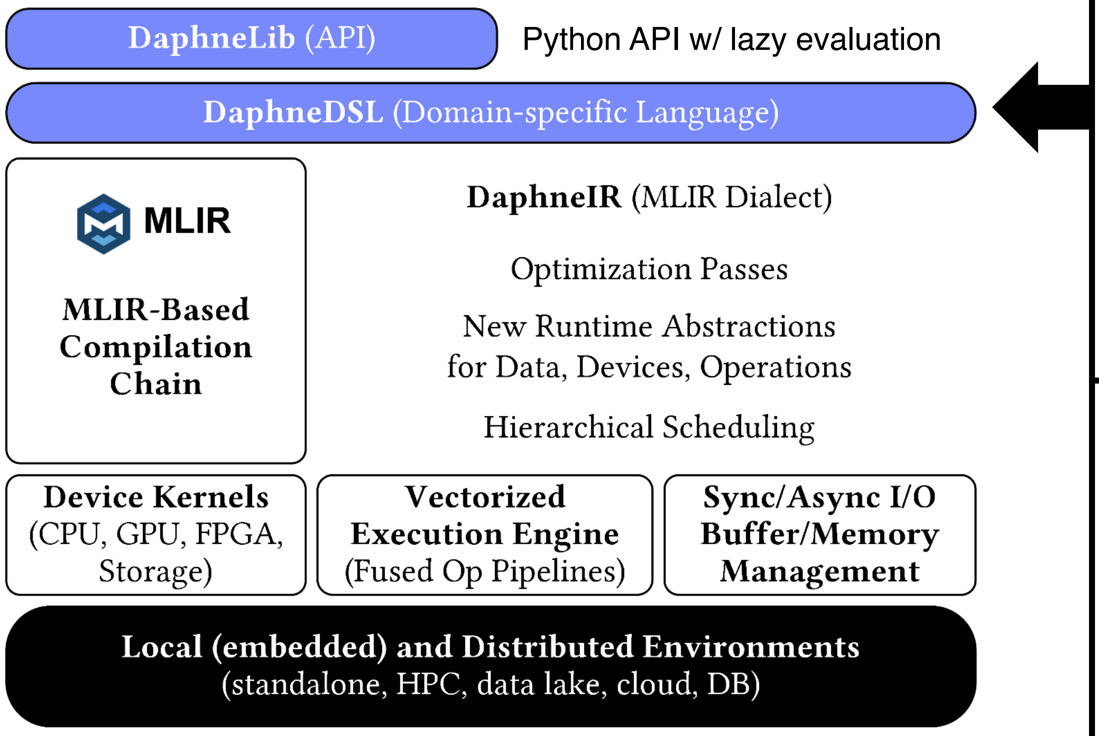
DAPHNE



Language Abstractions

DAPHNE

System Architecture



DSL for linear and relational algebra

- Coarse-grained **matrix/frame operations**
- **Physical data independence**

- Built-in operations for **linear and relational algebra**
- **High-level operations** (e.g., SQL, parameter servers, map)
- Conditional **control flow** (branches, loops)
- Typed and untyped **user-defined functions**

- **Hierarchy of primitives** for data science tasks

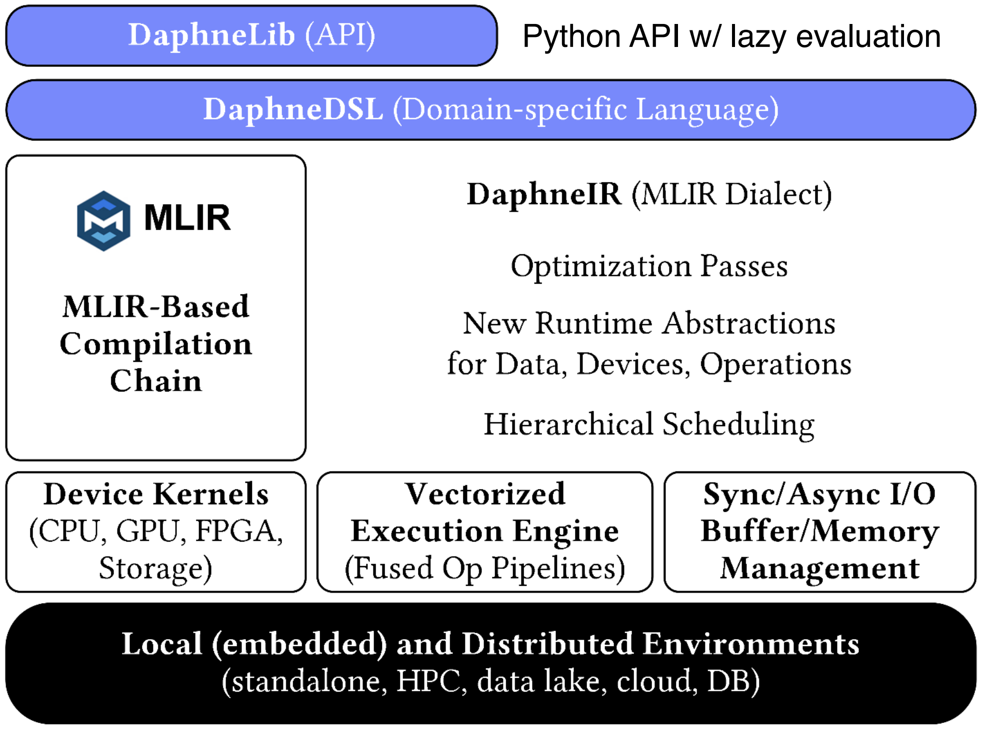
Example: linear regression model training (simplified)

```
def lm(X, y) { // X feature matrix, y labels
  colmu = means(X, 1);      // column means
  colsd = stddev(X, 1);     // column stddevs
  X = (X - colmu) / colsd; // shift and scale
  X = cbind(X, 1);          // append column of ones
  A = t(X) @ X;             // t for transpose
  b = t(X) @ y;             // @ for matrix mult
  return solve(A, b);       // system of linear eq
}
```

Optimizing Compiler

DAPHNE

System Architecture



MLIR-based Optimizing Compiler

- Intermediate representation **DaphneIR (MLIR dialect)**
- **Systematic lowering** from **domain-specific operations** to calls to pre-compiled **kernels for heterogeneous hardware**

- Traditional **programming language rewrites**
- Type & property **inference**, inter-procedural analysis
- **Domain-specific rewrites** from linear and relational algebra
- Memory management & garbage collection
- **Device placement** & **physical operator selection**

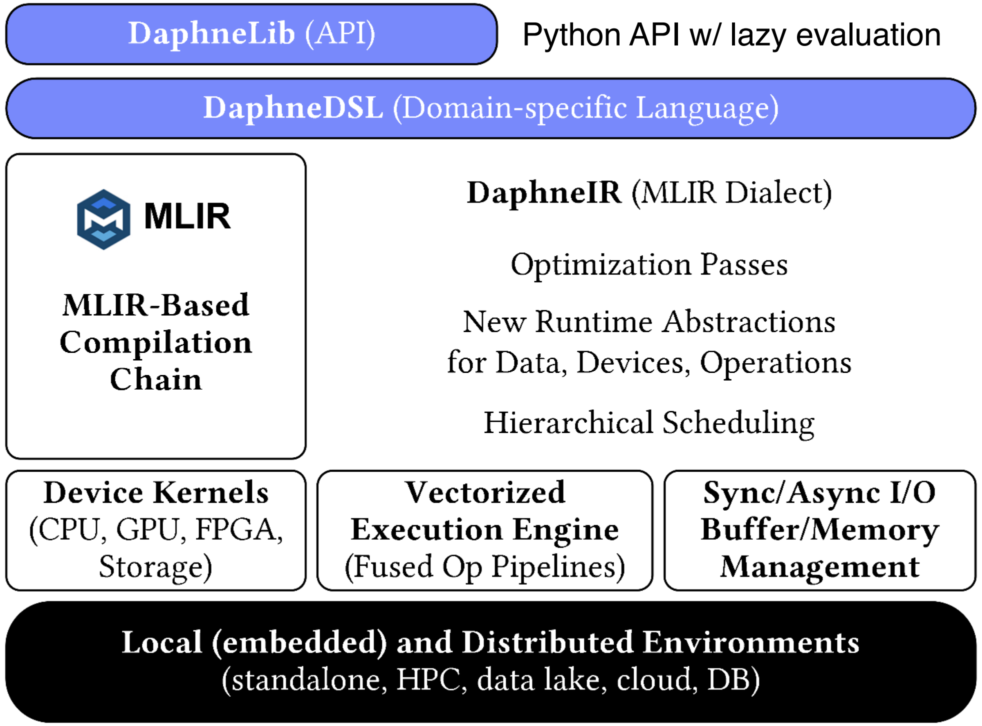
Example: linear regression model training (simplified)

```
%10:2 = "daphne.vectorizedPipeline"(%5, %colmu, %colsd, %7, %6) ({
^bb0(%arg0: ..., %arg1: ..., %arg2: ..., %arg3: ..., %arg4: ...):
  %12 = "daphne.ewSub"(%arg0, %arg1) : ...
  %13 = "daphne.ewDiv"(%12, %arg2) : ...
  %14 = "daphne.colBind"(%13, %arg3) : ...
  %15 = "daphne.gemv"(%14, %arg4) : ... // rewritten from matmul/@
  %16 = "daphne.syrk"(%14) : ...         // rewritten from matmul/@
  "daphne.return"(%15, %16) : ...
}, ...
```

Runtime

DAPHNE

System Architecture

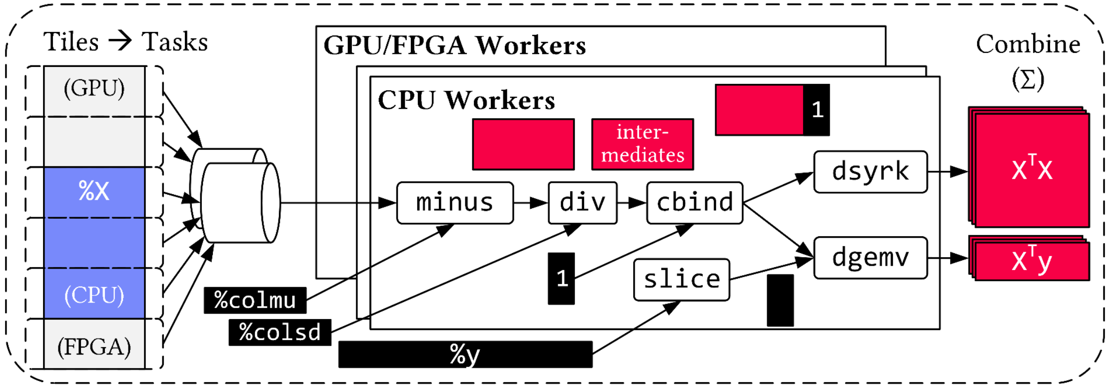


Distributed and Local Vectorized Execution

- Coarse grained tasks and cache-conscious data binding
- Fused operator pipelines on tiles/vectors of data
- Device **kernels for heterogeneous hardware**
- Integration of **computational storage** (e.g., eBPF programs)
- **Scheduling for load balancing** (e.g., for ops on sparse data)
- Different distributed backends (e.g., gRPC, OpenMPI)

Example: linear regression model training (simplified)

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



Holistic Extensibility

Challenges

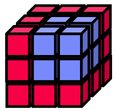
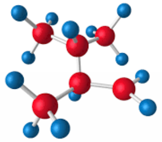
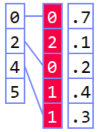
DAPHNE

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**
  - **End of Moore's law**
  - **Amdahl's law**

Increasing specialization

#1 Data Representations


graph



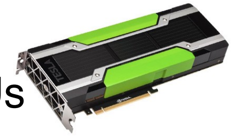
dense
sparse
compressed

Sparsity Exploitation from Algorithms to HW

#2 Data Placement

Local vs **distributed**


CPUs/ NUMA

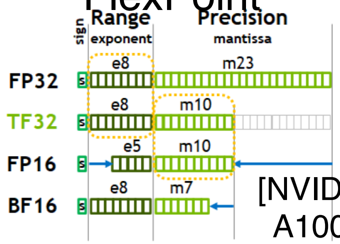

GPUs



FPGAs/ ASICs

#3 Data (Value) Types

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



[NVIDIA A100]

Holistic Extensibility

DAPHNE

- **Every relevant aspect of a system for IDA pipelines should be extensible by the user without a deep understanding of the system**

3 main aspects

Operators

- New logical operations
- New physical operators with dedicated code for each target hardware device

Data representation

- Often co-designed with hardware & algorithms/apps
- New data types (layouts)
- New value types

Optimization/scheduling

- Optimizer must be able to reason about extensions
- New optimizer passes
- New runtime schedulers

- **Research challenges**
  - How to balance expressiveness vs. increased system complexity?
  - How to achieve superb performance underneath abstractions?

Low Barrier of Entry

DAPHNE

- Typically, all three aspects need to interact to fully integrate a novel hardware device
- **But: low barrier of entry required for user adoption**
  - No unnecessary effort
  - No unnecessary restrictions
  - Ideally no need to touch target system's code base
- Facilitate **exploratory specialization**

- **Examples**
  - Add single physical operator for accelerator, but not entire engine
  - New data/value type should work with existing physical operators (with acceptable performance), no need for re-implementation
  - No requirement for deep integration into optimizer (allow usage via hints for initial experiments)

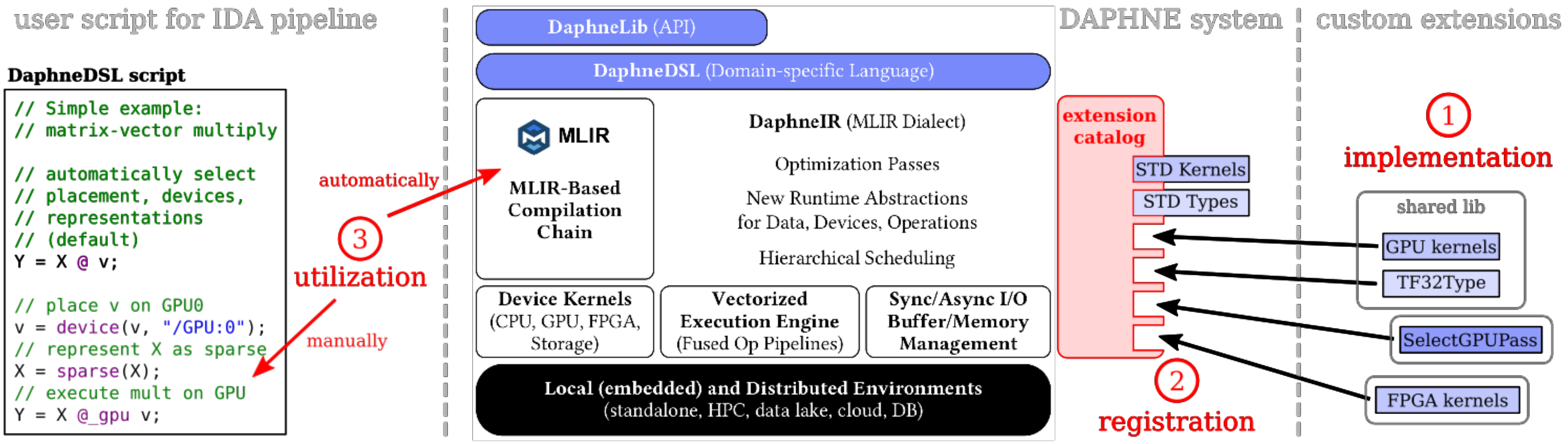
Towards Holistic Extensibility in DAPHNE

3-Step Approach for Adding/Using Extensions

DAPHNE

Abstractions for cost models
Interesting properties

Balance of expressiveness and complexity
Efficiency underneath abstractions



Propagation of hints

Optimize access to extension catalog

- Building upon existing works on extensible **DBMSs** from **1980/90s**
- Tailored to needs of **today's** data processing **systems for IDA pipelines**

[Michael J. Carey, Laura M. Haas: Extensible Database Management Systems. **SIGMOD Rec. 19(4) 1990**]

Adding a New Physical Operator

- **1 Implementation**
  - C++ function, operation-specific interface
  - Lots of freedom inside the implementation
  - Offering APIs for common tasks (data transfer, memory management, …)
- **2 Registration**
  - Provide basic information to make DAPHNE compiler aware of new kernel
  - Optionally more information (interesting properties/traits, cost models, …)
- **3 Utilization**
  - Automatically through multiple dispatch based on data/value types
  - automatically through cost models
  - Manually in DaphneDSL

Planned extensions (examples)

- Kernels for various hardware backends
- Readers/writers for various file formats
- Readers/writers for special storage hardware

C++

```
void myMatMul(
  DenseMatrix<float>& res,
  const DenseMatrix<float>* lhs,
  const DenseMatrix<float>* rhs
) {
  // e.g., cublasSgemm(...), etc.
}
```

myKernels.so

Operation	daphne::MatMulOp
Func name	**myMatMul**
Shared lib	myKernels.so
Backend	GPU
Input types	[DenseMat<float>, DenseMat<float>]
Output types	[DenseMat<float>]
…	…

extension catalog

DaphneDSL

```
// Fully automatic (default)
C = A @ B;

// Choose exact kernel
C = $myMatMul(A, B);
// Choose backend
C = A @_GPU B;
```

Adding a New Data/Value Type

Data

- **1 Implementation**
  - Interface of DAPHNE matrix or frame (get/set/append values, slice, …)
- **2 Registration**

Logical type	Frame
Type name	ArrowFrame
Shared lib	myDataTypes.so
…	…

- **3 Utilization**
  - Automatically via cost models (e.g., physical size, access patterns, …)
  - Manually via hints/casts in DaphneDSL

```
// Choose data representation
AF = as.ArrowFrame(F);
```

⇒ **Associated operations as ordinary kernels** (print/parse, cast, read/write from/to file)

Value

- **1 Implementation**
  - New C++ type (e.g., struct) with interface for interplay with DAPHNE data types
- **2 Registration**

Type name	Quantized8
Size	8 bit
…	…

- **3 Utilization**
  - Explicitly by the user (application semantics)

```
XQ = as.matrix<Quantized8>(X);
```

  - Internal selection by the system (e.g., runtime-accuracy trade-off)

Planned extensions (examples)
- Data types for data layouts of various libraries
- Data types for various sparse matrix formats
- Value types for quantized, packed, complex, …

Extending the Optimizer or Scheduler

Optimizer

- **1 Implementation**
  - Leverage extensible nature of MLIR-based optimizing compiler
  - Implement new MLIR pass
- **2 Registration**
  - Use MLIR-provided means
  - Also: integration of 3rd-party dialects from the MLIR ecosystem
- **3 Utilization**
  - Configure DAPHNE optimizer chain to use new pass at the right stage

Planned extensions (examples)
- Passes employing new extensions
- Passes deciding placement on accelerators
- Extensions for propagation/estimation of interesting data properties

Scheduler

- **1 Implementation**
  - Also follow specific interface
- **2 Registration**

Name	Guided Self-Scheduling (GSS)
Type	dynamic

- **3 Utilization**
  - Manual selection as default scheme

```
/bin/daphne --scheduler GSS myScript.daph
```

  - Manual selection for a part of a script

```
[scheduler="GSS", nthreads=32] {
  // complex calculations
}
```

Planned extensions (examples)
- Various static/dynamic self-scheduling techniques for local multi-threaded and distributed execution

Summary

Summary

DAPHNE

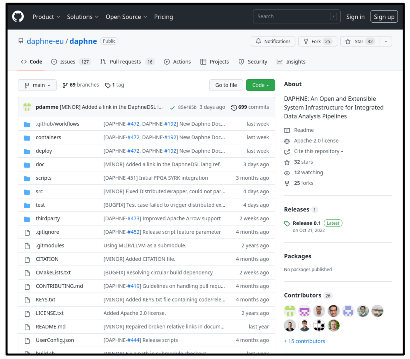
- **Holistic Extensibility for Systems tailored to today's IDA Pipelines**
  - To address increasing specialization
  - Low barrier of entry for adoption by researchers and users
- **Current Status**
  - Running DAPHNE prototype
  - Ongoing work in various components
  - Extensibility features: WIP
- **Why should you care?**
  - Simple integration of prototypes (e.g., for novel hardware)
  - Easy experimentation for researchers
  - **Open for collaborations**

Integrated Data Analysis (IDA) Pipelines

DM + ML + HPC

DAPHNE overall objective:
Open and extensible system infrastructure

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



Open source (Apache v2 license)
https://github.com/daphne-eu/daphne
v0.2 released on July 31, 2023

Towards inclusive developer community

DAPHNE

Feel free to get in touch:
patrick.damme@tu-berlin.de

https://daphne-eu.eu/

https://github.com/daphne-eu

https://twitter.com/daphne_eu

https://www.linkedin.com/in/daphne-eu-project-695735230/

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