intel labs

LAVA
An open-source software framework for neuromorphic computing

Andreas Wild, Mathis Richter
Intel Neuromorphic Computing Lab

Open Neuromorphic
May 31, 2023

ONM

intel Neuromorphic Research Community

Legal Information

Performance varies by use, configuration and other factors. Learn more at www.Intel.com/PerformanceIndex.

Performance results are based on testing as of dates shown in configurations and may not reflect all publicly available updates. See backup for configuration details. No product or component can be absolutely secure.

Your costs and results may vary.

Results have been estimated or simulated.

Intel technologies may require enabled hardware, software or service activation.

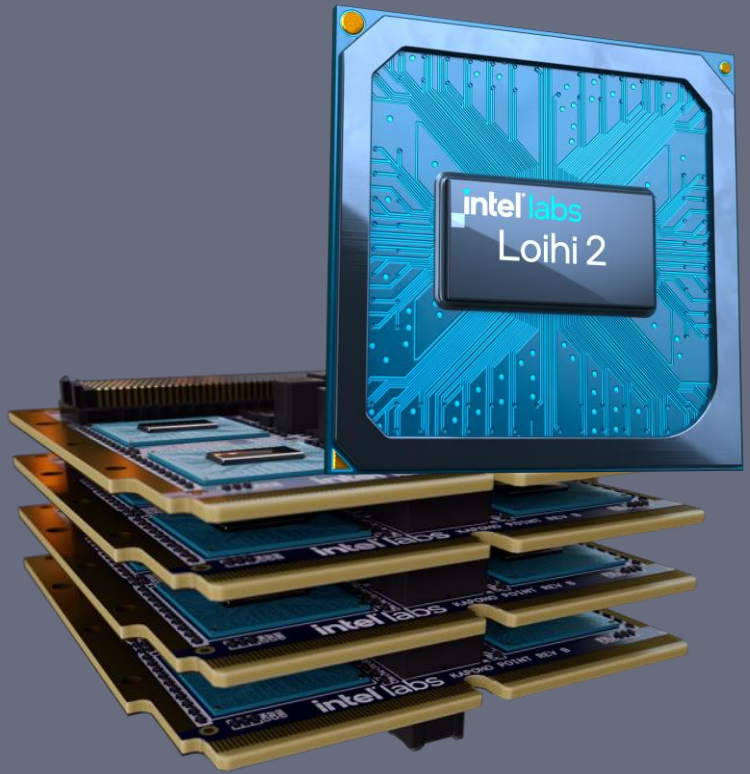
Intel does not control or audit third-party data. You should consult other sources to evaluate accuracy.

Intel disclaims all express and implied warranties, including without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement, as well as any warranty arising from course of performance, course of dealing, or usage in trade.

What is Lava?

And why did Intel launch it?

A new SW framework sparked by the arrival of Loihi 2



- Up to 10x faster processing capability*
- Up to 60x more inter-chip bandwidth*
- Up to 1 million neurons with 15x greater resource density*
- Programmable neurons
- Graded spikes
- 3-Factor learning
- 10G Ethernet I/F to host

LAVA

An open-source software framework for neuromorphic computing

* specs and configuration details can be found at intel.com/neuromorphic

What is Lava?

- Full SW stack from runtime, to compiler, to algorithm/application libraries
- Brain-inspired programming model for heterogeneous HW
  - Parallel & asynchronous
  - Event-based computation/communication
- Seeded by Intel but open-source and increasingly community-driven

LAVA

Event-based communication

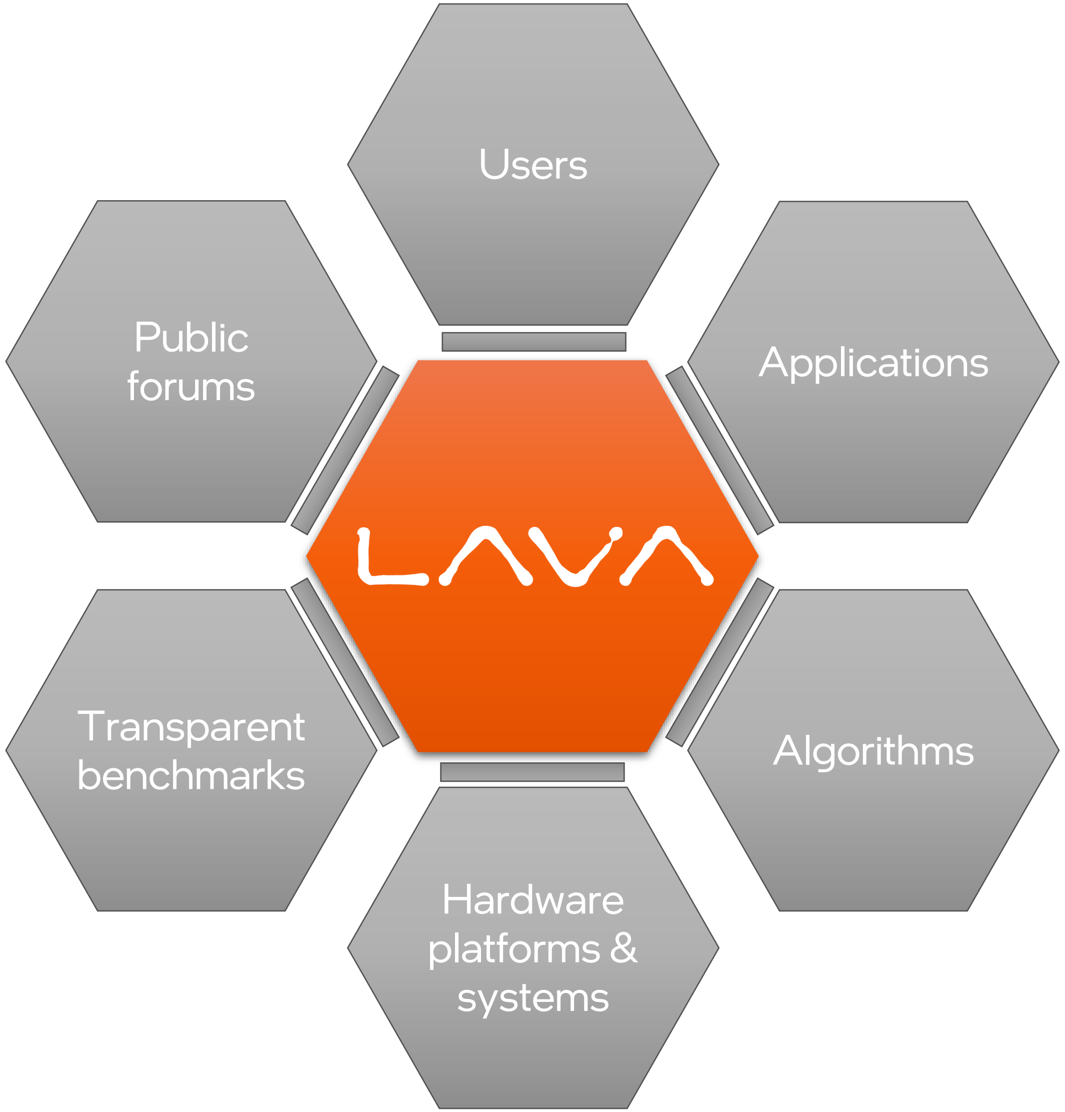
Multi-Paradigm

Multi-Abstraction

Multi-Platform

Why Lava?

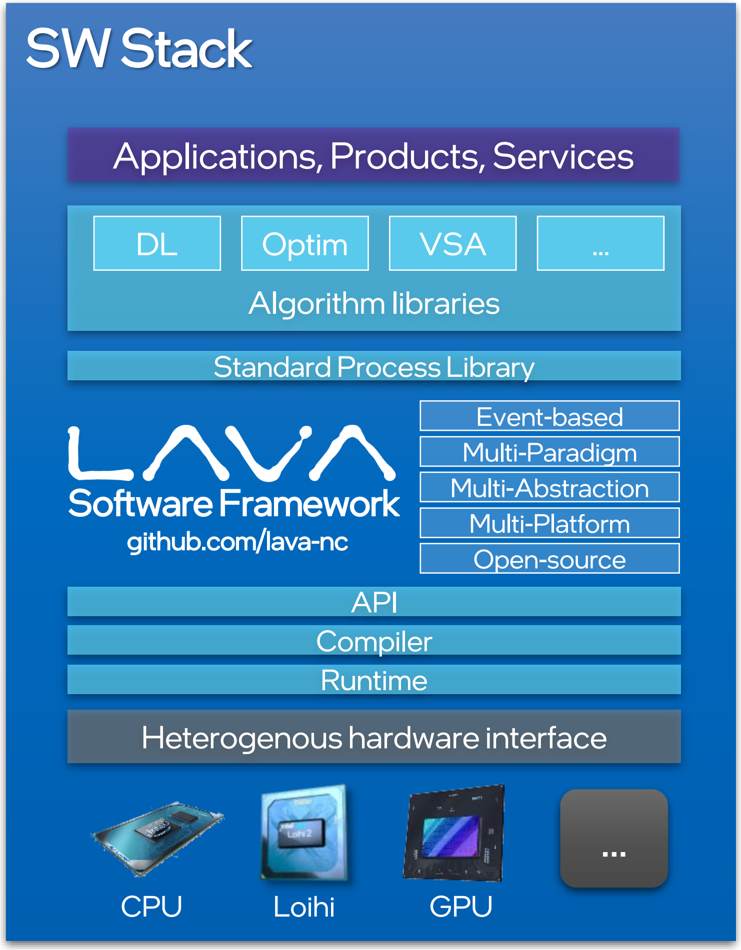
- Converge neuromorphic SW development towards open standard
- Make *exotic* neuromorphic systems accessible to non-expert developers
- Accelerate adoption of neuromorphic technologies
- Enable orders of magnitude gains in compute efficiency



Capabilities

What you can do with Lava today

Lava SW stack

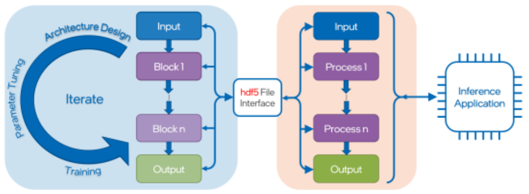


Capabilities today	Platform
▪ Cross-platform compiler/runtime	CPU/Loihi 2
▪ Programming in Python & C	CPU/Loihi 2
▪ "Arbitrary" neuron models via μCode programming	Loihi 2
▪ Synaptic plasticity via 2/3-factor learning rules	CPU/Loihi2
▪ Power, performance, activity, memory profiler	Loihi 2
▪ Lava-dl: Direct training and model deployment	CPU/Loihi 2
▪ Lava-optim: Solvers for QUBO and QP problems	CPU/Loihi 2
▪ Lava-dnf: Connectivity generators for attractor networks	CPU/Loihi 2
▪ Comprehensive documentation and tutorials	CPU/Loihi 2

Lava algorithm libraries

lava-dl

- Direct & HW-aware training of event-based DNNs
- Rich neuron model library (feed-forward & recurrent)

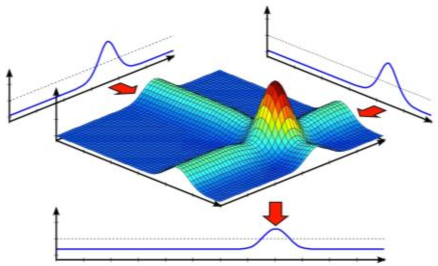


lava-optim

- Family of constraint optimization solvers
- Today: QP, QUBO, LCA, BO
- Future: MPC, ILP, ...
- Standalone use or as part of AI applications

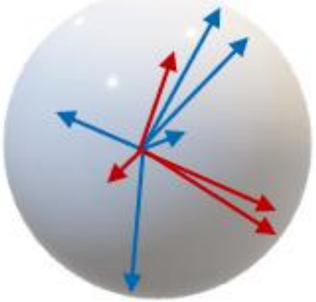
lava-dnf

- Design models with attractor dynamics
- Stabilize temporal data
- Selective data processing
- Dynamic working memories



lava-vsa (WIP)

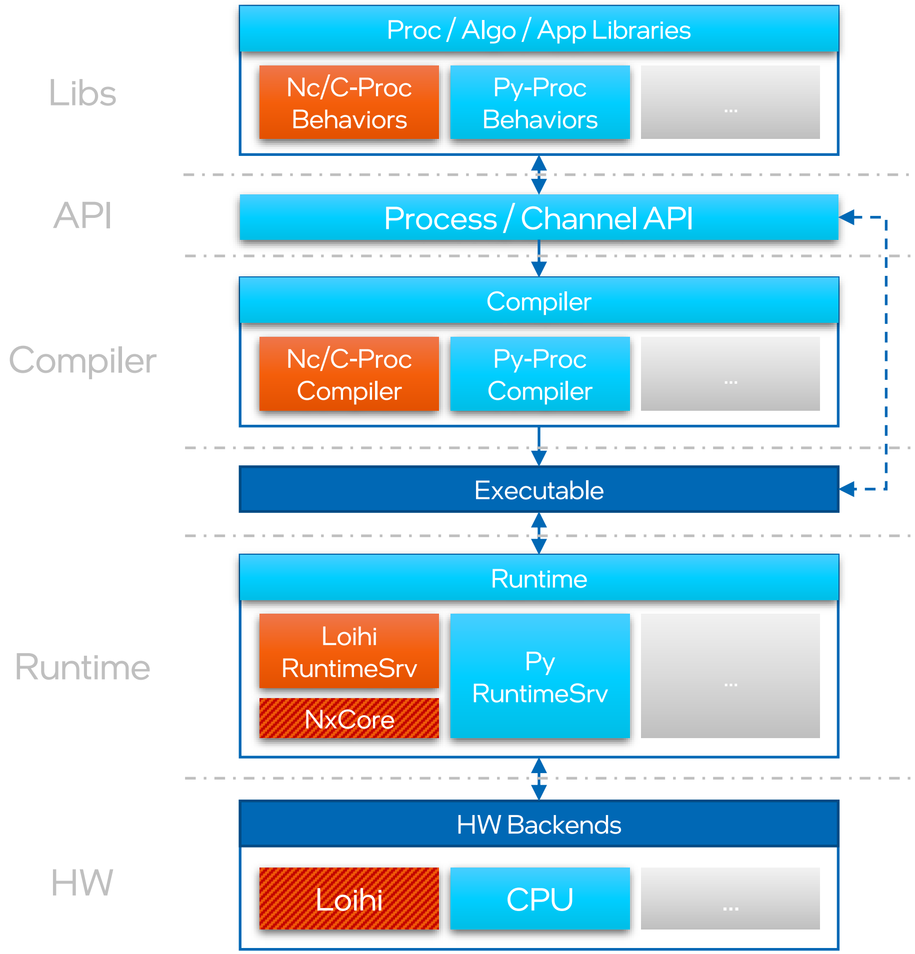
- API for algebraic model description for VSAs
- Library of data types and operations (composition, binding, factorization, ...)



Future directions

- lava-io (sensor/actuator interfaces)
- lava-robotics (control, planning, physical simulator interfaces)
- lava-evolve (evolutionary training methods)
- lava-ui (graphical network creation, visualization, debugging)
- Signal processing
- Off-the-shelf apps (segmentation, tracking, keyword detection, ...)
- Neural simulators (Brian2Lava, ...)

Lava stack & open sourcing



Framework components

Lava Framework with CPU backend:
- Open-source
- Code distributed via GitHub
- End user documentation/tutorials
- Python

Lava Loihi backend:
- Proprietary
- For INRC members or selected partners
- Developer documentation
- Mostly Python, some C

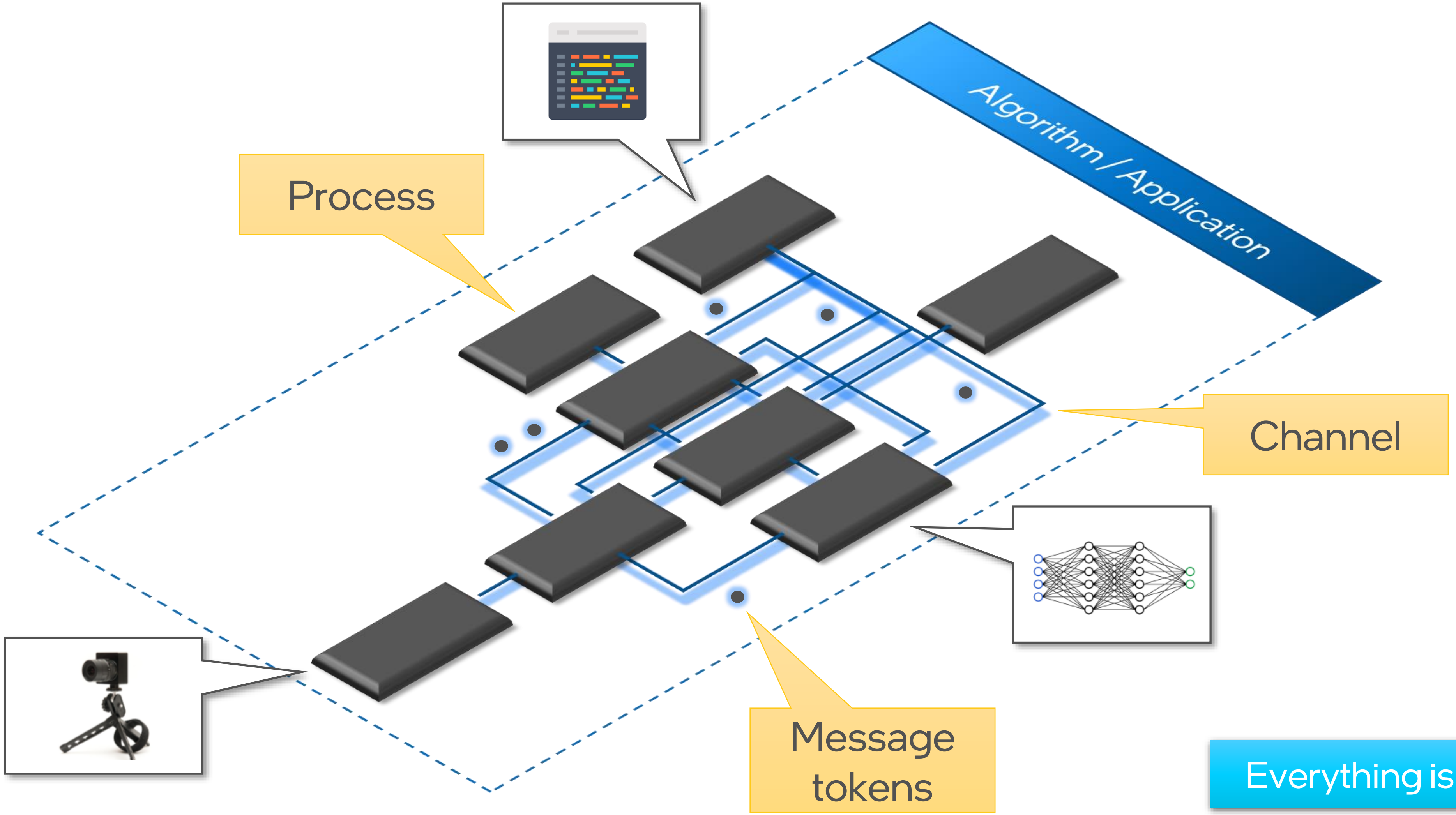
Loihi + NxCore:
- Proprietary
- For INRC members or selected partners
- Developer documentation/tutorials
- Python / C / C++

Using Lava

Basic Lava concepts

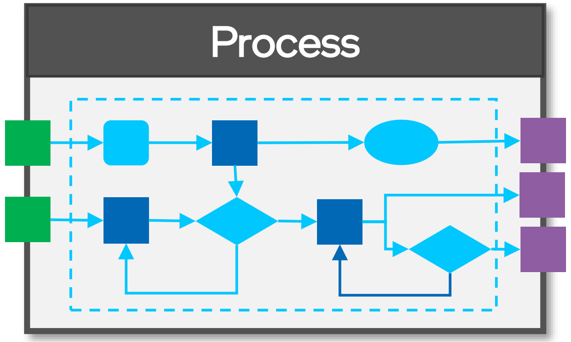
Processes and channels

Lava's fundamental building blocks



Everything is a Process in Lava

Processes



Input Ports Internal Vars Output Ports
Logic

- Implementation-agnostic API
- InPorts/OutPorts for message-based communication via channels
- Internal state & behavior

Example Process definition:

```
class Proc(AbstractProcess):
    def __init__(self, **kwargs):
        super().__init__(**kwargs)
        # Input ports
        self.inp1 = InPort(shape=(2, 1))
        ...
        # Output ports
        self.out1 = OutPort(shape=(10, 1))
        ...
        # Internal variables
        self.var1 = Var(shape=(1, 2), init=10)
        var2_init = kwargs.get("var2_init", 0)
        self.var2 = Var(shape=(1, 1), init=var2_init)

    def an_api_method(self, *kwargs):
        ...
```

All Processes derive from AbstractProcess

__init__ defines Vars and Ports

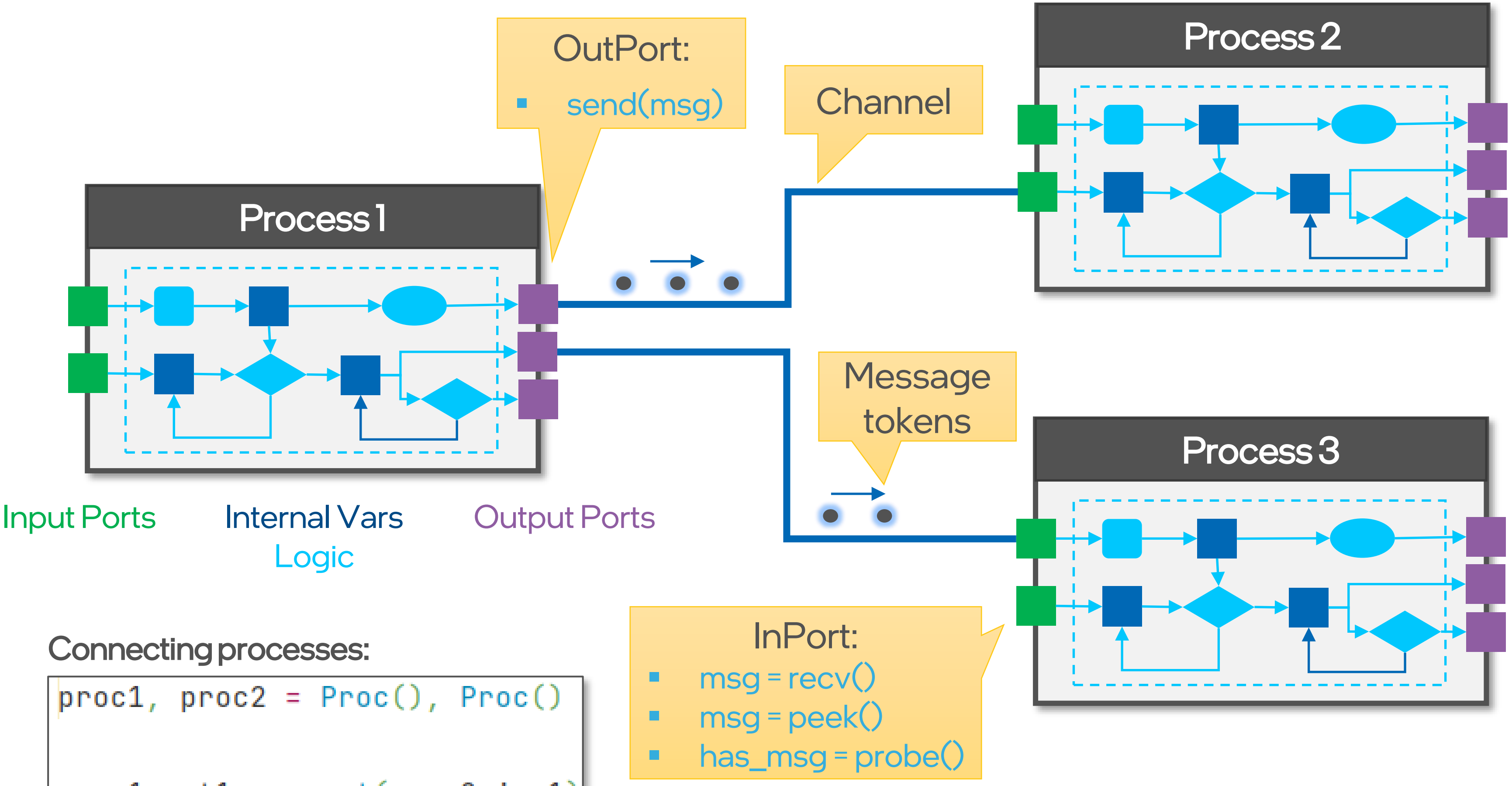
Behavioral implementation defined separately…

Tensor-valued Vars and Ports

Additional API methods

Processes provide implementation-agnostic API but separate behavioral implementation

Processes and channels



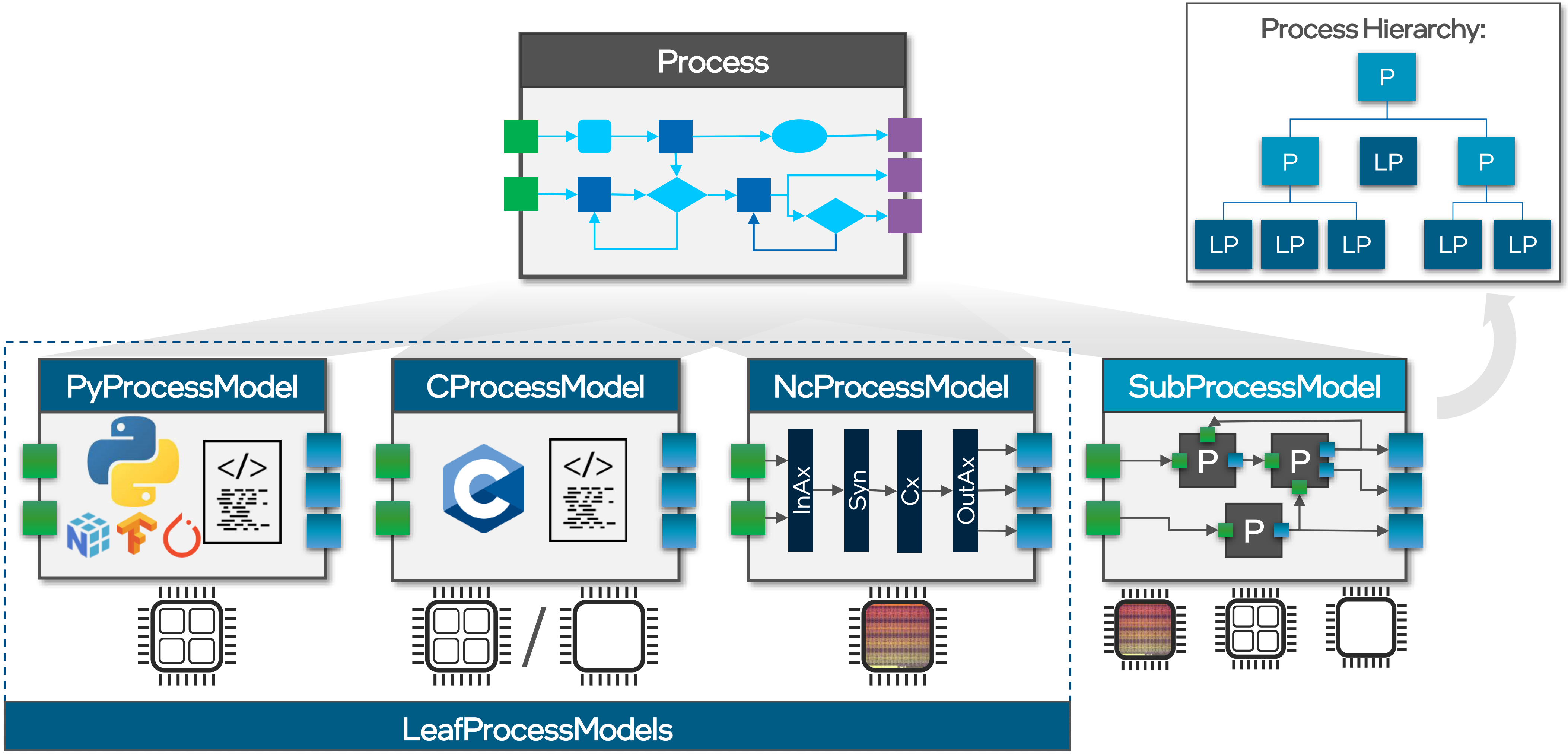
Connecting processes:

```
proc1, proc2 = Proc(), Proc()

proc1.out1.connect(proc2.inp1)
```

Processes communicate through channels via message tokens

Behavioral models → ProcessModel



Two classes of ProcModels: LeafProcModels vs. hierarchical SubProcModels

Examples

Examples

- Tour through Lava
  - lava / tutorials / end_to_end / tutorialX00_tour_through_lava.ipynb
- Low level tutorial library
  - lava / tutorials / end_to_end , in_depth
- Lava-dl: PilotNet
  - lava-dl / tutorials / lava / lib / dl / slayer / pilotnet / train.ipynb
- Lava-optimization: QUBO
  - lava-optimization / tutorials / tutorial_02_solving_qubos.ipynb

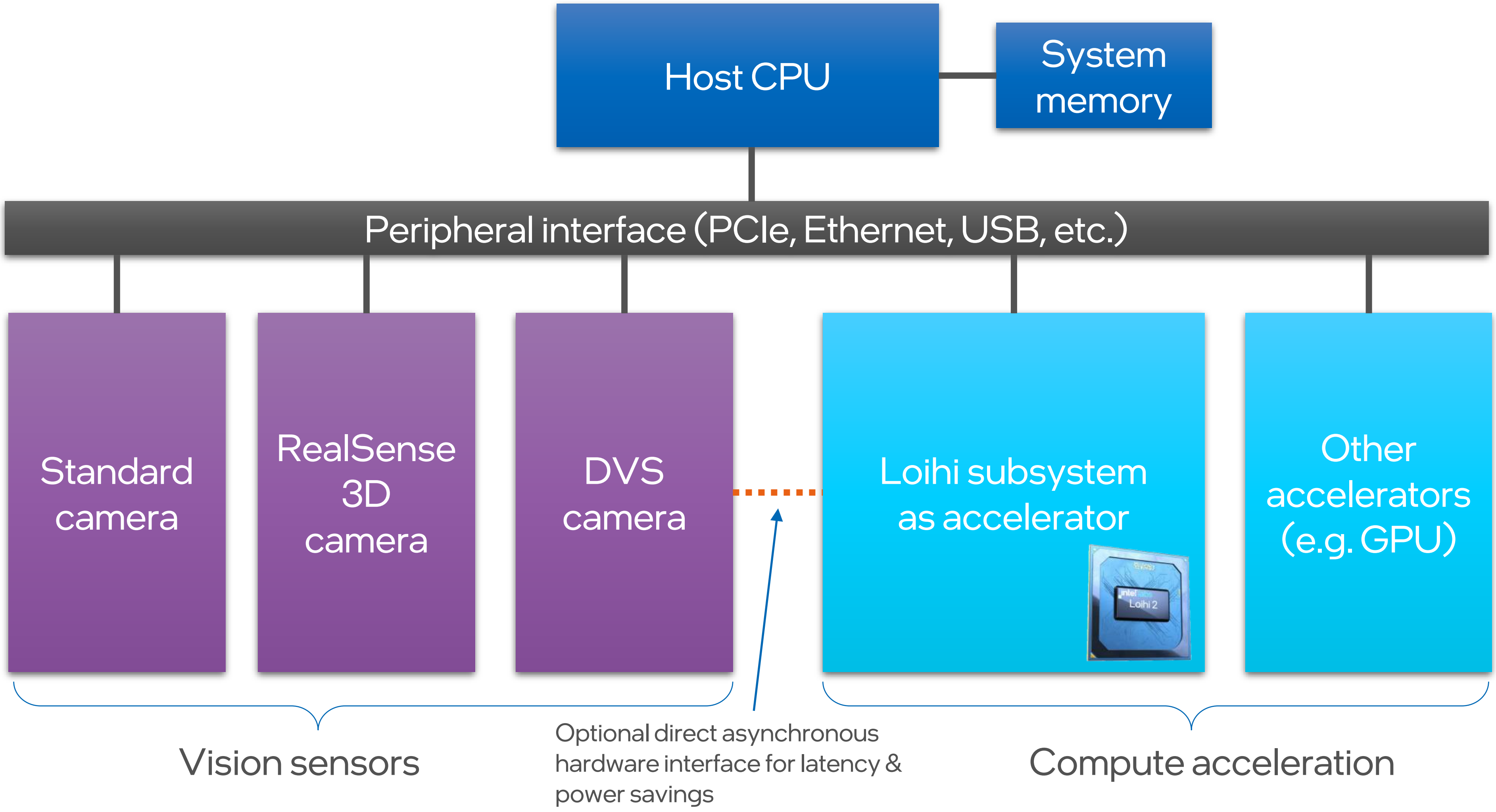
Current development highlights

New SW infrastructure and applications

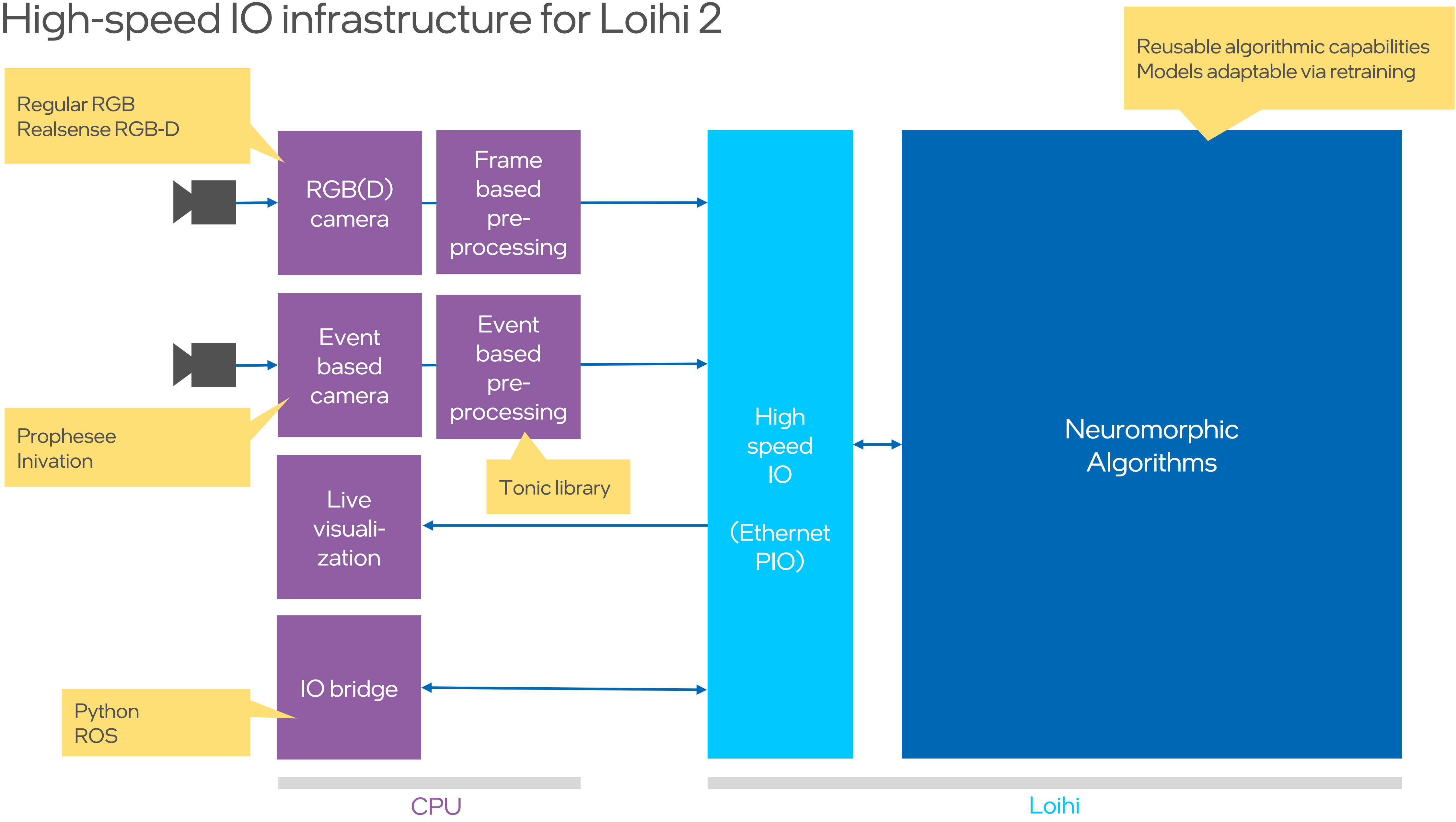
Latest Lava release

- Lava 0.7 – Minor feature additions (April 2023)
  - Tutorial for performance profiling
  - State probes
  - Synaptic delays
  - Bugfixes
- Preparations for larger feature release
  - Sparse compression of connectivity
  - Multiple dendritic accumulators
  - Support for real-time vision capabilities

General Loihi system architecture

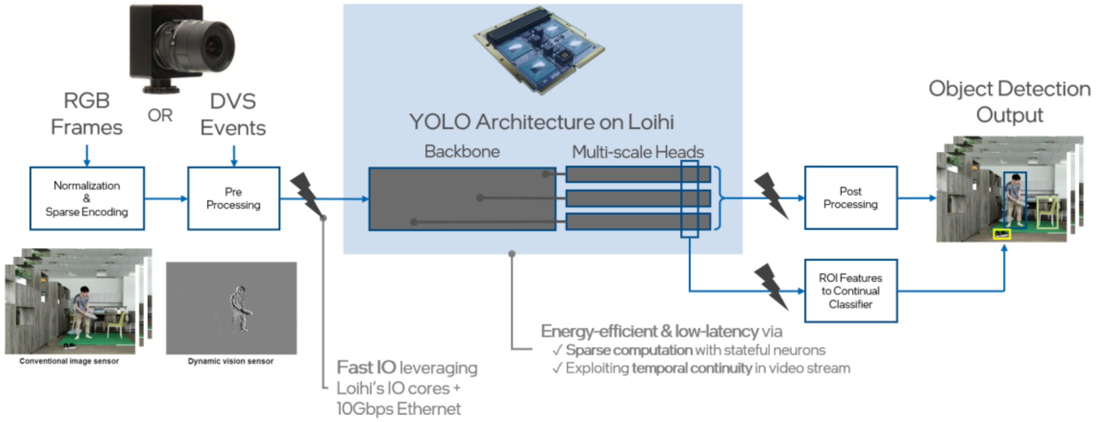


High-speed IO infrastructure for Loihi 2

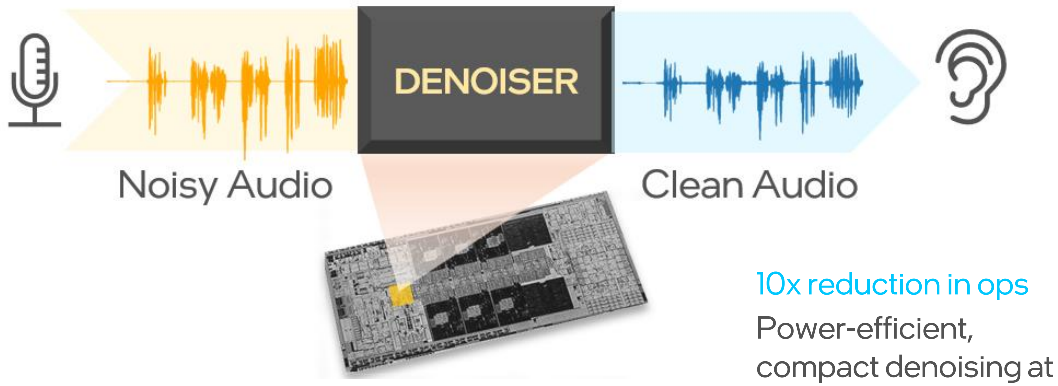


Upcoming signal processing application examples

DNN-based object detection & localization



Audio denoising

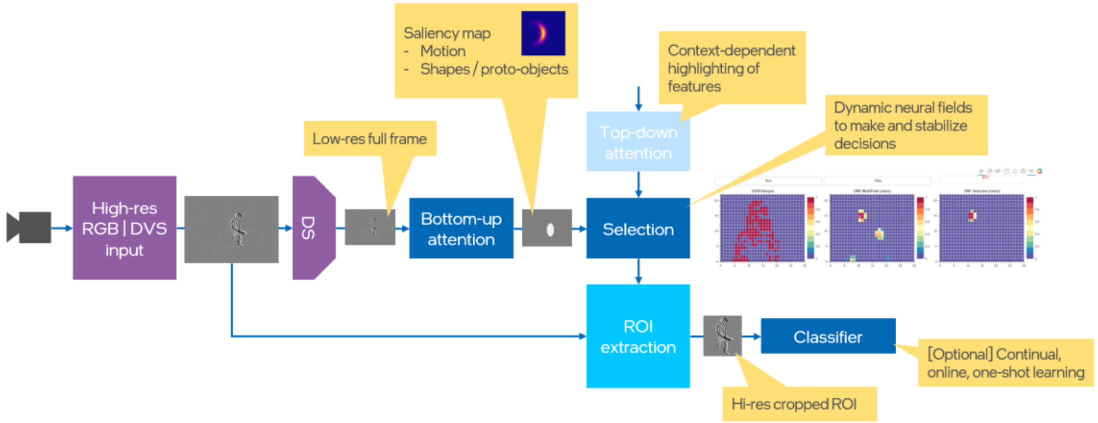


10x reduction in ops

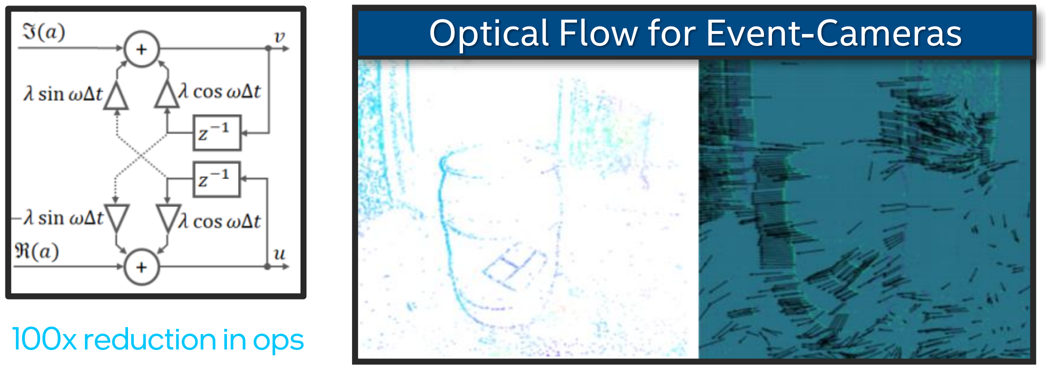
Power-efficient, compact denoising at comparable quality

Timcheck et al, Intel DNS challenge, arXiv:2303.09503

Feedback-driven attention object detection



Optical flow with RF neurons



100x reduction in ops

G. Orchard et al, "Efficient Neuromorphic Signal Processing with Loihi 2"

New online/on-chip learning in Lava

Learning rules as sum-of-products:

$$Z_l(t) = Z_l(t - t_E) + \sum_{r=1}^{N_P} D_r^h \left(S_r \cdot \prod_{s=1}^{N_F} F_{l,r,s}^f \right)$$

Synaptic state variables

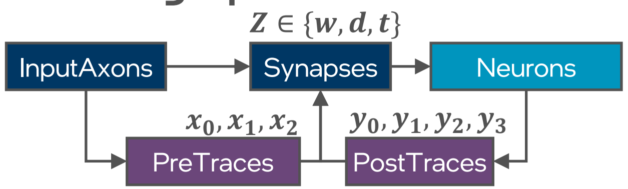
Learning epoch

Dependency operator/factor

Scaling factor

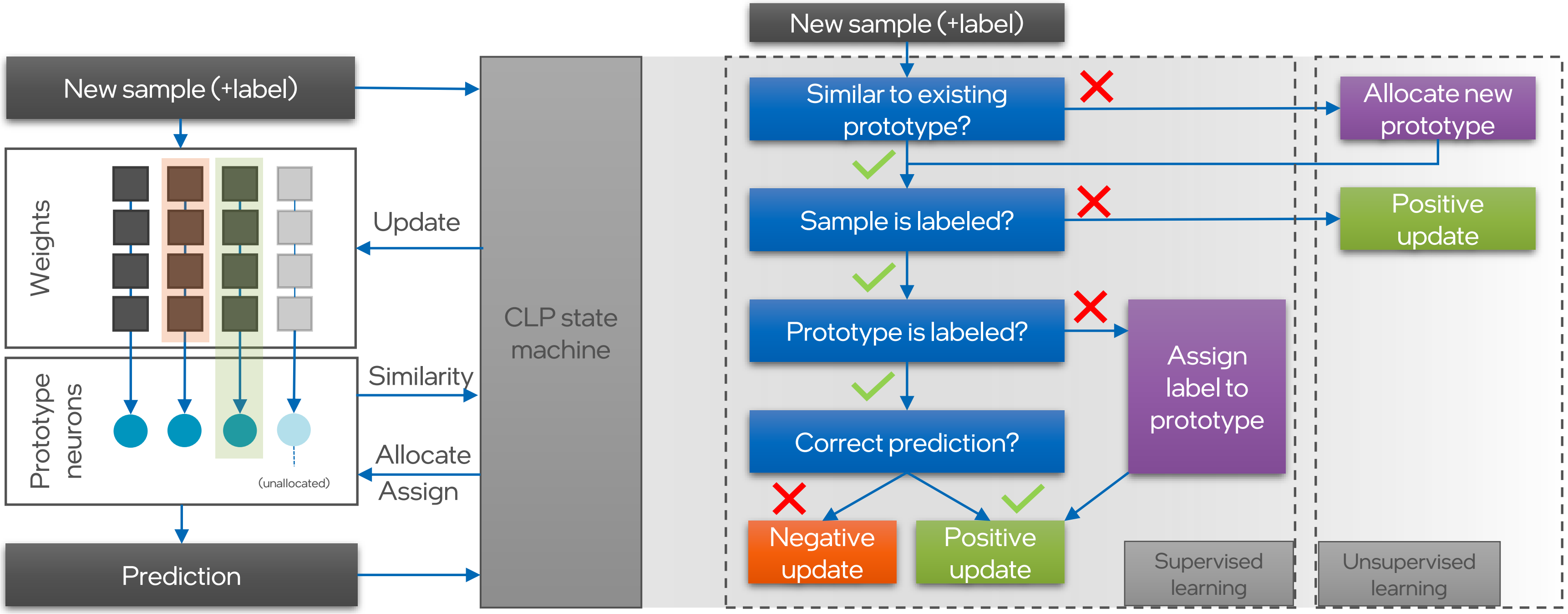
Factor

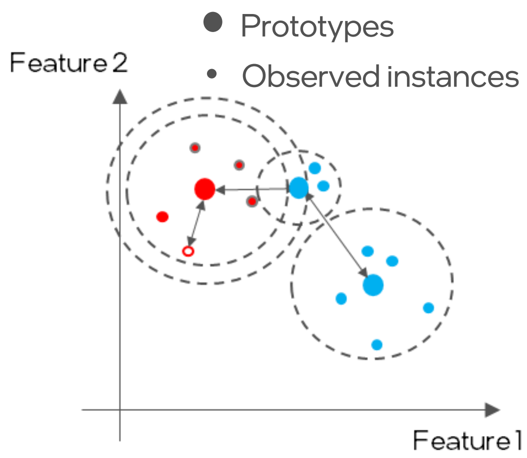
Computational graph:



- Learning supported on CPU/Loihi 2
- Local, 3F-online/on-chip learning
- Currently applied to:
  - Continual Learning Prototype classifier
  - Differentiable neural plasticity

Generic *Continual Learning Prototype* classifier





Characteristics:

- Single-layer, local updates
- Interpretable
- Adjustable memory capacity
- Performant & energy efficient

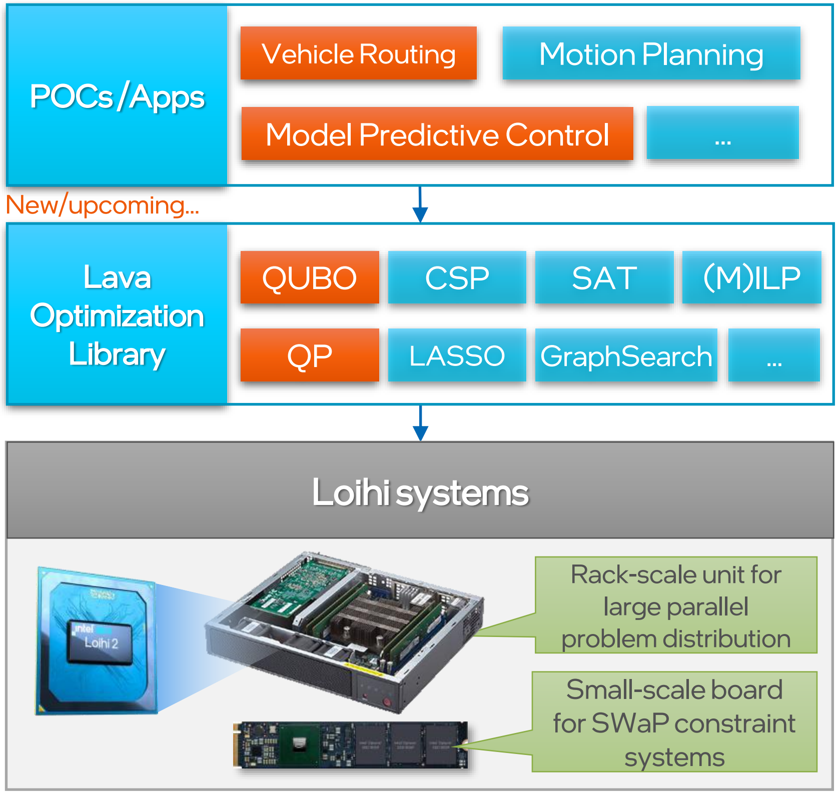
Capabilities:

- Novelty detection
- One-shot learning
- Continual online learning
- Open-set recognition

Enhanced constraint optimization solvers and POCs

Solving optimization problems with orders of magnitude gains in speed and power

Lava-Optimization Library



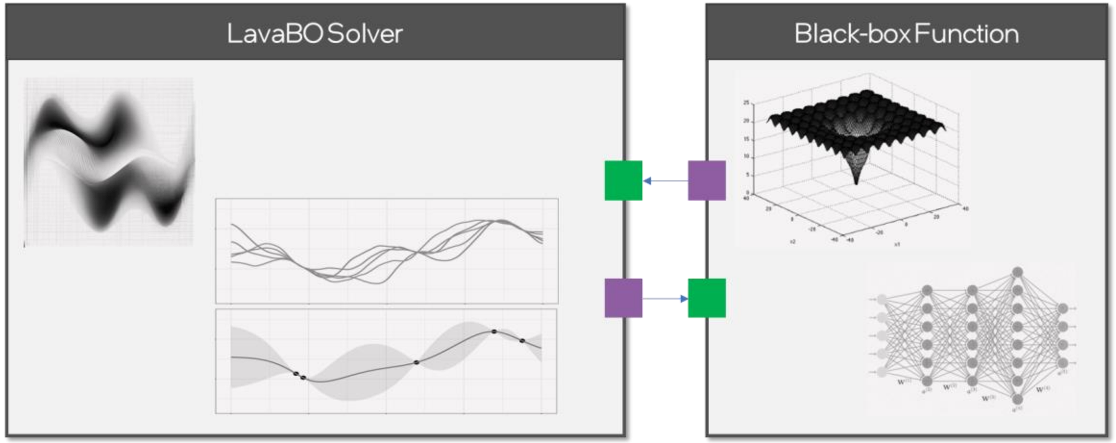
Capabilities:

- Up to 10^5 × EDP gains on Loihi 2 vs. SotA CPU solver
- Scalability from edge to cloud
- Productive HW-agnostic API

Community projects

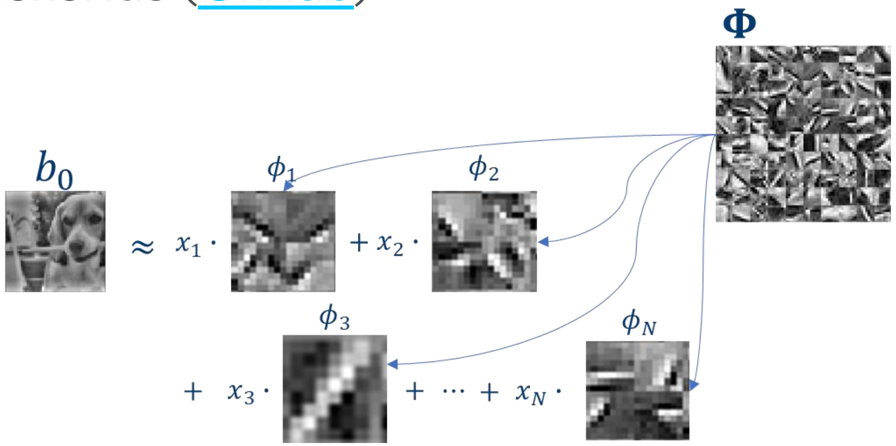
Bayesian Optimization
(George Mason University)

- Objective: Optimize (hyper-) parameters of functions that are non-convex or expensive to evaluate (e.g., entire deep networks)
- Features: Supports three acquisition functions, two acquisition estimators, flexible search space specification
- Status: Solver supports CPU backend; Loihi 2 WIP (Github)



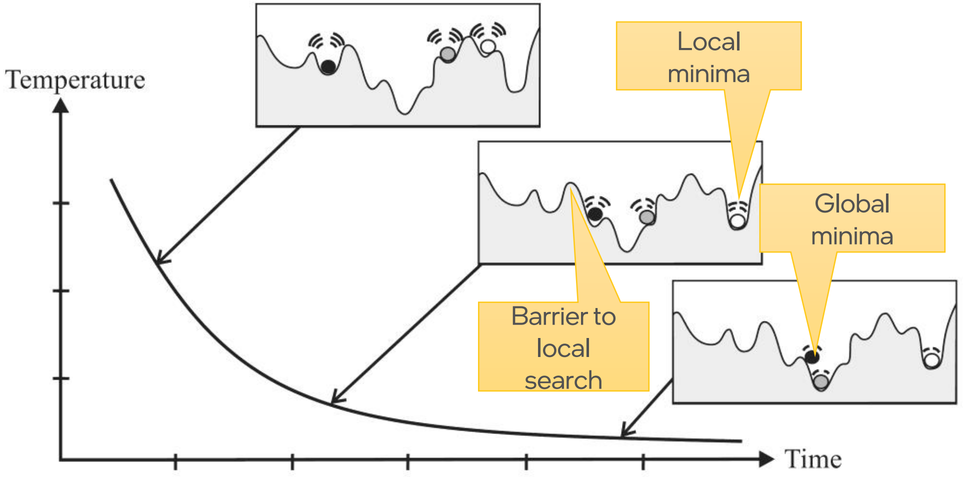
LASSO Optimization
(Pacific Northwest National Lab)

- Objective: Minimize $\frac{1}{2}||\mathrm{b}_0 - \Phi\mathrm{x}||_2^2 + \lambda||\mathrm{x}||_1$, with known feature set Φ to get optimal sparse representations x of input b_0.
- Features: Uses locally competitive algorithm for efficiency. Known to be up to 10^4 times more energy efficient on Loihi than CPU.
- Status: Solver supports CPU & Loihi 2 backends (Github)



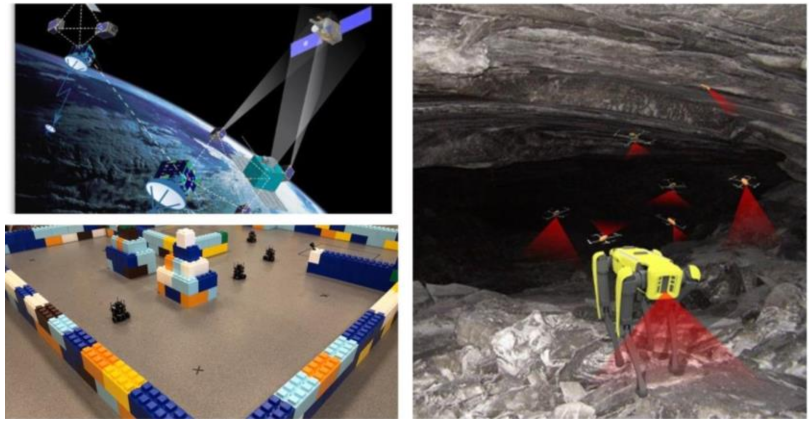
Simulated Annealing
(MIT, ORNL, U. Bonn)

- Objective: Accelerated combinatorial optimization for problems of currently intractable complexity at CERN
- Features: explores overall search space before narrowing down towards global optimum
- Status: POC working on Loihi 2 backend; benchmarking data collected from D-Wave



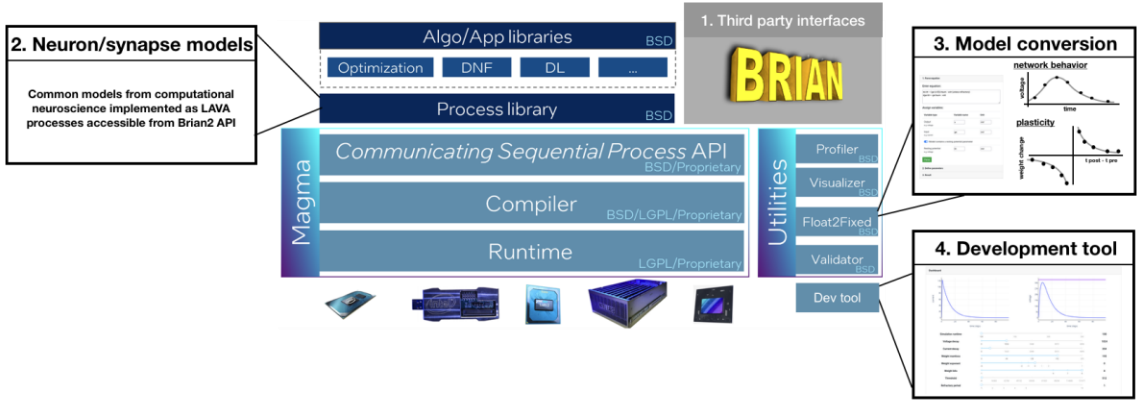
Model Predictive Control
(Lulea University)

- Objective: Model predictive control for planning of swarms of drones and quadrupeds
- Features: builds control algorithms on top of Lava's QP solver, derives stability and convergence guarantees for the solvers under real-time latency constraints. Build demos with physical robots.
- Status: stability analysis for PIPG/MPC algorithm, control loop with Lava's QP solver. CPU for now, Loihi 2 WIP.



Brian2Lava
(U. Goettingen)

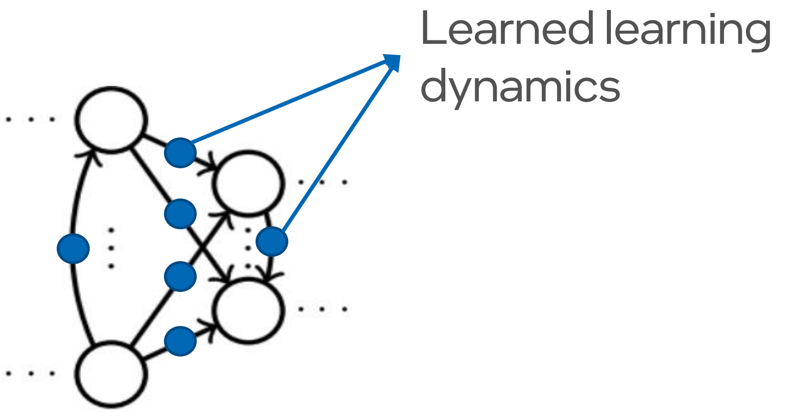
- Objective: Brian 2 interface for Lava to deploy brain-inspired algorithms
- Feature: Automatic conversion of Brian models to Lava/Loihi 2 + Models + Utilities
- Status: Solver supports CPU backend; Loihi 2 WIP



https://brian2lava.gitlab.io/

Differentiable plasticity
(RWTH Aachen, FZ Juelich, NRL)

- Objective: Backpropagating to learn rapid real time weight adaptation on chip
- Features: Gradient through learning dynamics + Learned (via backprop) learning rule coefficients
- Status: Lava-DL supports for DNNs + differentiable local learning rules



Discussion

https://intel-ncl.atlassian.net/wiki/spaces/INRC/overview intel®