Toward A High-Performance Emulation Platform for Brian-Inspired Intelligent Systems

-Exploring Dataflow-Based Execution Model and Beyond

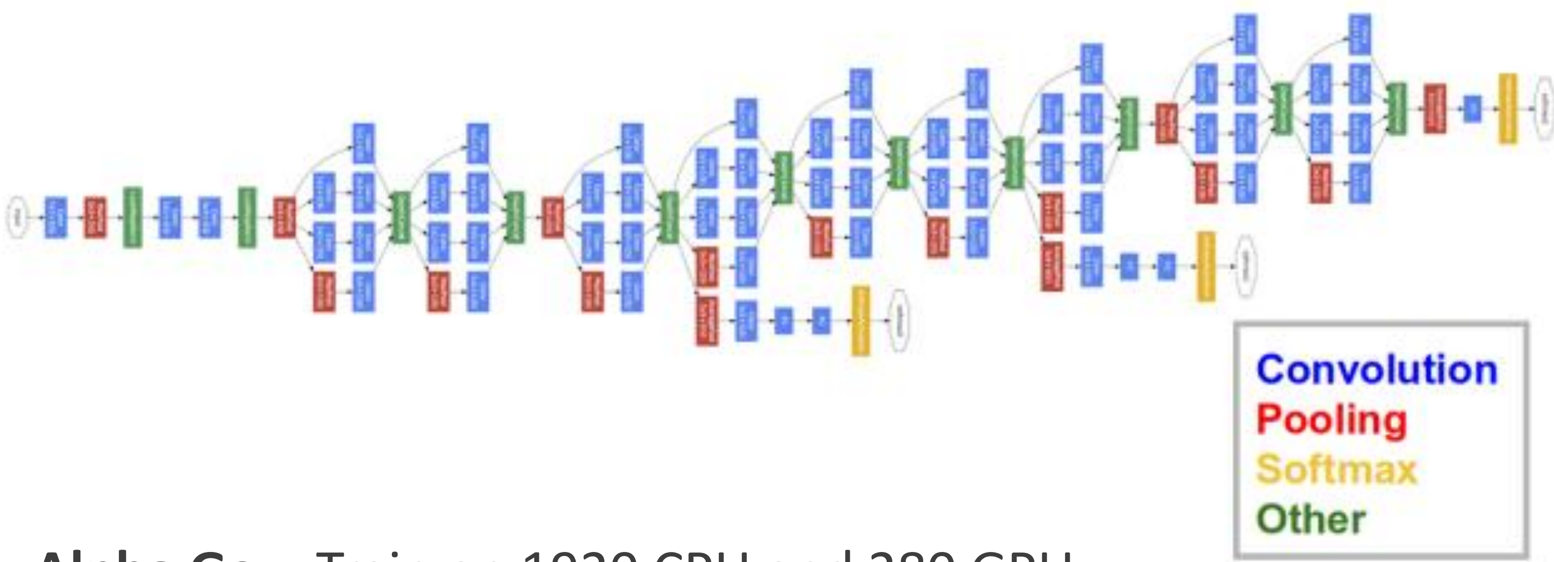
Sihan Zeng

Presenter: Jose Monsalve Diaz

Siddhisanket Raskar

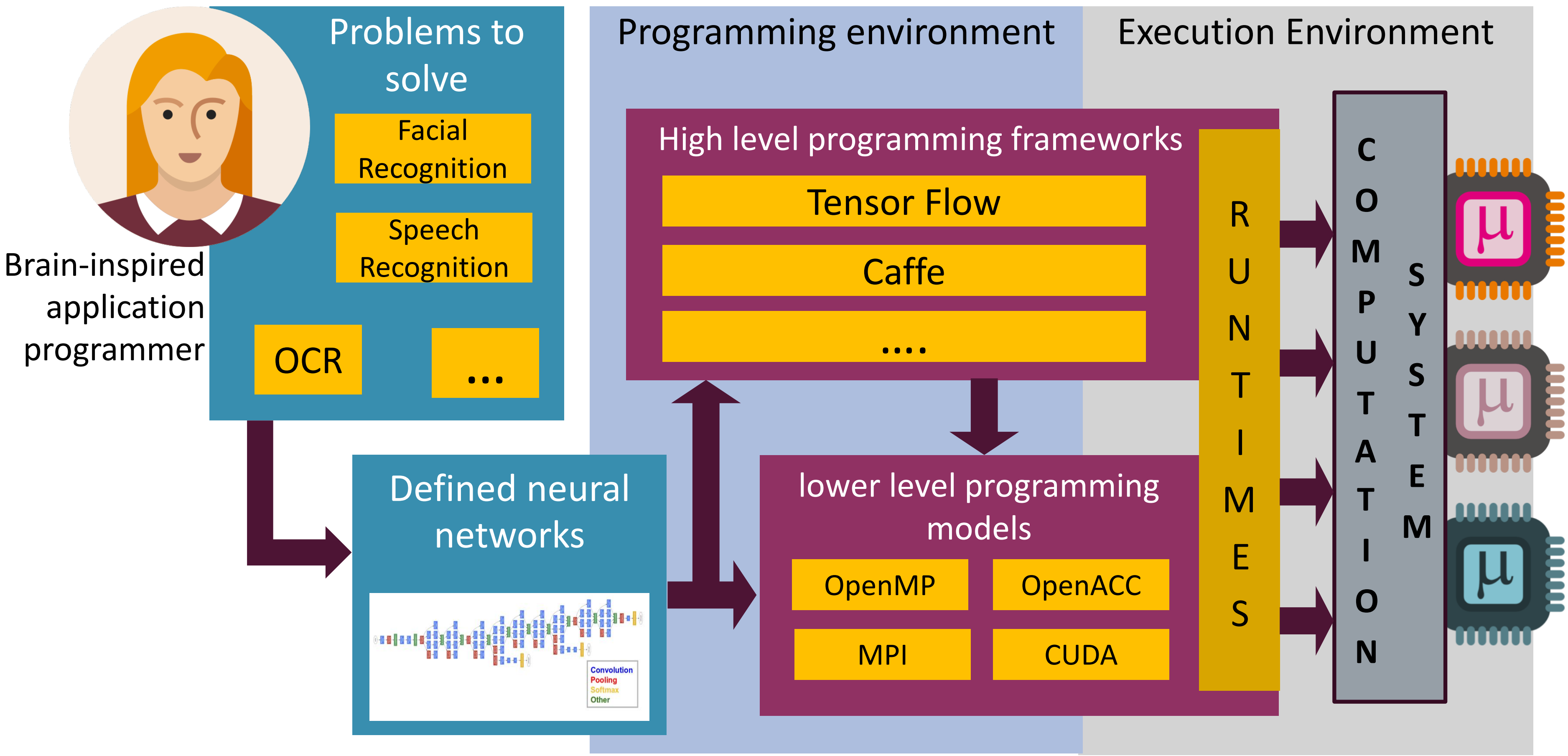
Intelligent System

- **GoogLenet** - 22 layers, 5M parameters.



- **Alpha Go** - Train on 1920 CPU and 280 GPU.
  - How to program neural networks on multi-core or heterogeneous system allowing efficiency and scalability?

Neural networks



Motivation

- Currently, there is a lack of common abstraction in parallel computational systems
  - Each programming model provides their vision of the machine
  - Poor interoperability between different programming frameworks
  - Lack of hardware support for such abstractions result in a large variety of software implemented runtime systems
    - Extra effort is required to inter-operation of those frameworks

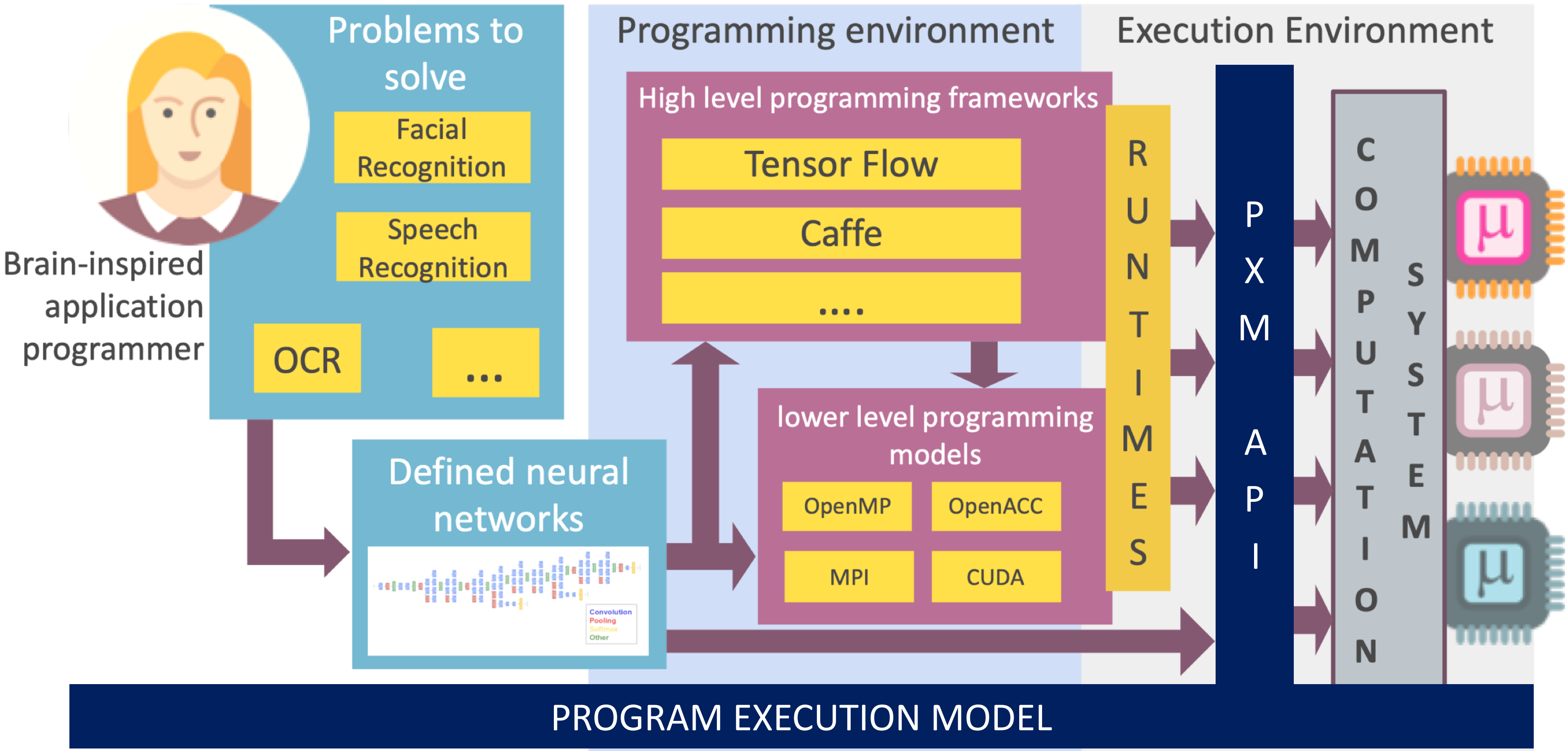
Program Execution Models

The **program execution model (PXM)** is the basic **low-level abstraction of the underlying system architecture** upon which our programming model, compilation strategy, runtime system, and other software components are developed. The PXM (and its API) serves as an **interface between the architecture and the software**.

Program Execution Models

Unlike an instruction set architecture (ISA) specification, which usually focuses on lower level details (such as instruction encoding and organization of registers for a specific processor), the **PXM refers to machine organization at a higher level for a whole class of high-end machines as viewed by the users**

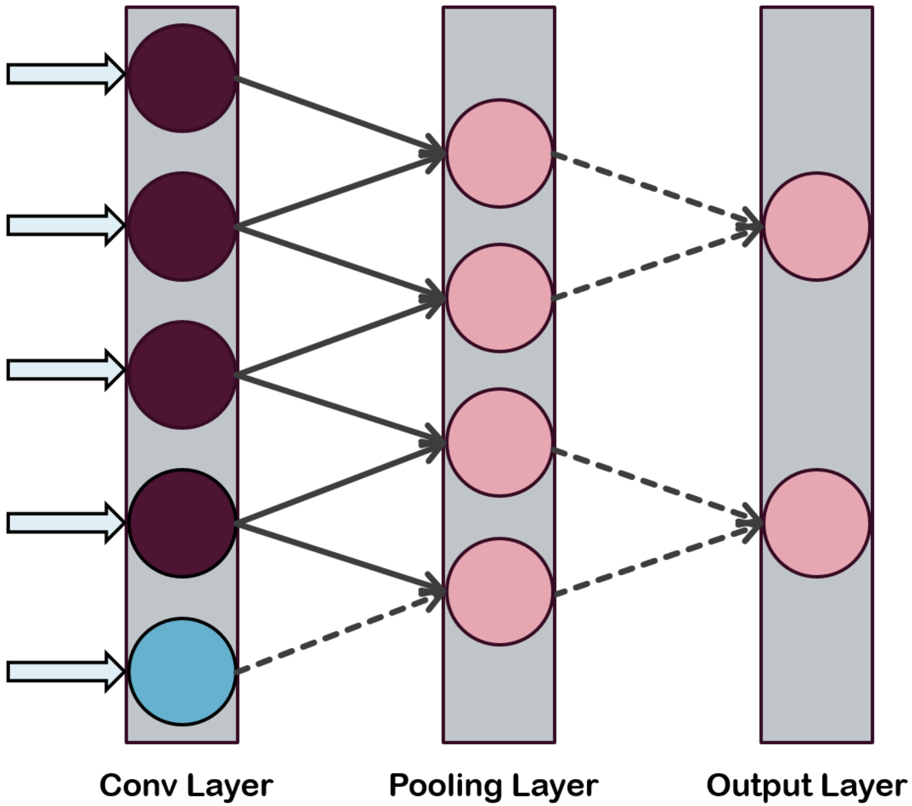
Neural networks



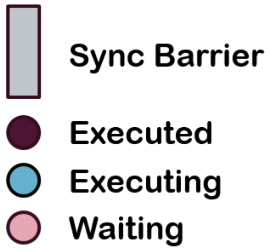
Motivation

- Existing simulation platforms (Tensorflow, Caffe) adopts bulk synchronous parallel models in many cases (e.g. OpenMP , MPI) which suffers from several drawbacks:
  - Inefficient synchronization
    - Leading to improper utilization of resources
  - Poor scalability
  - Problematic for irregular problems (e.g. Brain-inspired algorithms)

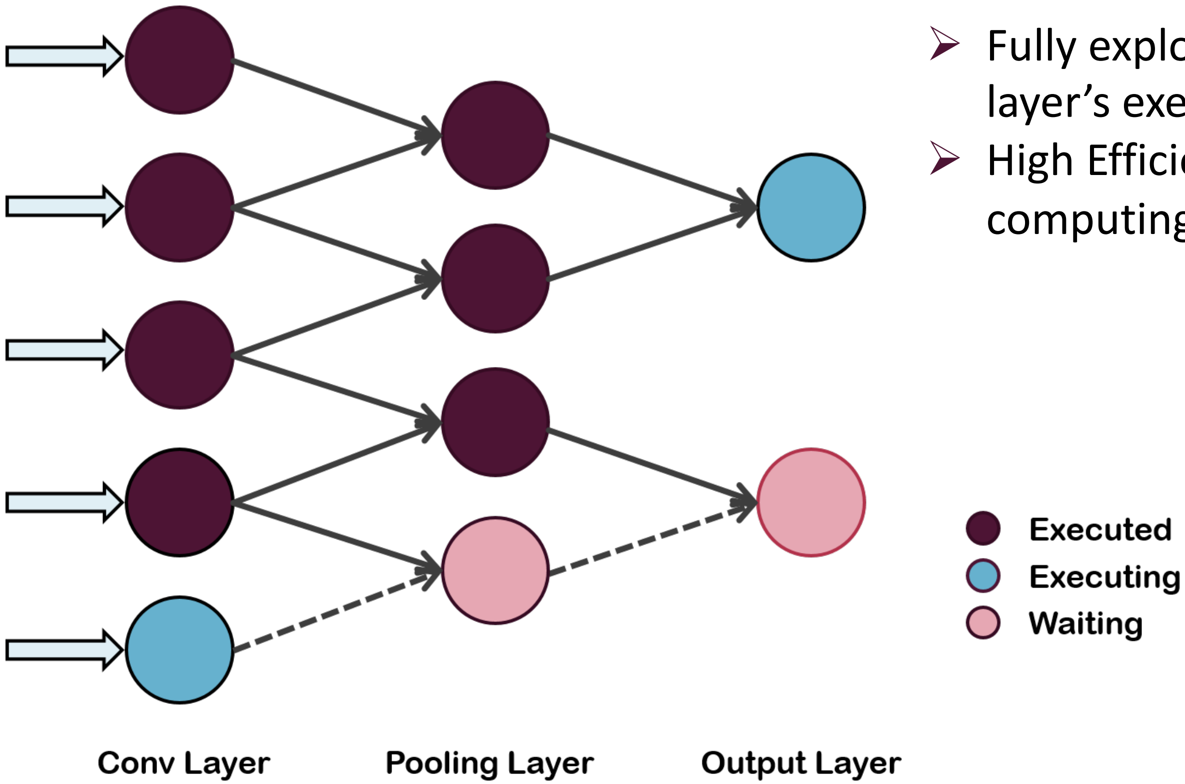
Coarse grain programming model



- Every unit waits until all neurons finish
- Unpredictable and variable execution time of neurons, specially for complex hardware



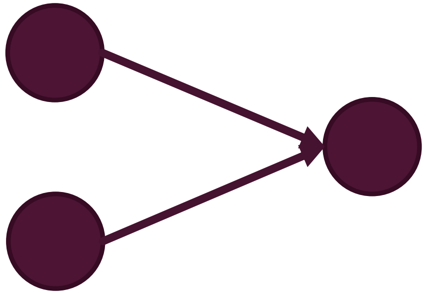
Fine grain programming model



- Fully exploit hardware through layer's execution overlapping
- High Efficiency on multicore computing systems – Scalability

Dataflow model of computation

- Computation is expressed in terms of operations and their dependencies
  - No program counter
- Operations are scheduled for execution as soon as their dependencies are satisfied and the needed resources are free
- Reduces the drawbacks of coarse grain programming models and provides a new path for large scale parallel computing
  - Asynchronous
  - Activated upon input availability



Objective

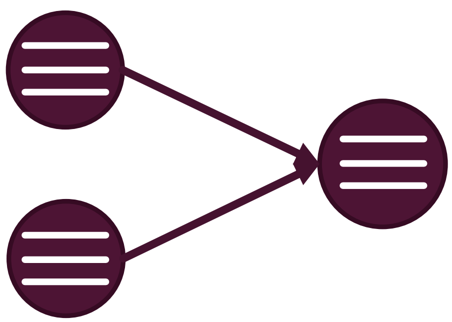
- Provide an argument in favor of dataflow-based programming model as a possible solution for the challenge that brain inspired intelligent system faces when deployed on exascale systems with manycore architectures
- Develop a fine grain simulation system for brain inspired computing (neural network) on multi-core/heterogeneous system, to further demonstrate the feasibility and superiority of our proposal.

Codelet Model

-A fine grain asynchronous program execution model

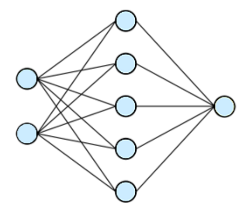
Codelet Model

- Inspired by **Dataflow Architectural Model** and **Von Neumann Architectural Model**
  - Combining the benefits of both worlds
- Asynchronous execution of fine-grain event-driven codelets
- Contextualized grouping of codelets into asynchronous threaded procedures
  - Locality of data and computation

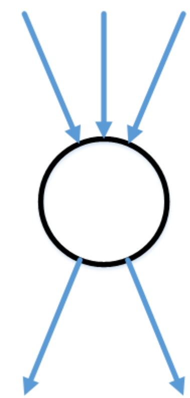


Codelet Model

- Program are defined as Direct Acyclic Graphs called **<u>Codelet Graphs</u>**
  - Nodes in the graph are codelets (Computation tasks)
  - Edges in the graph are data (or control) dependencies



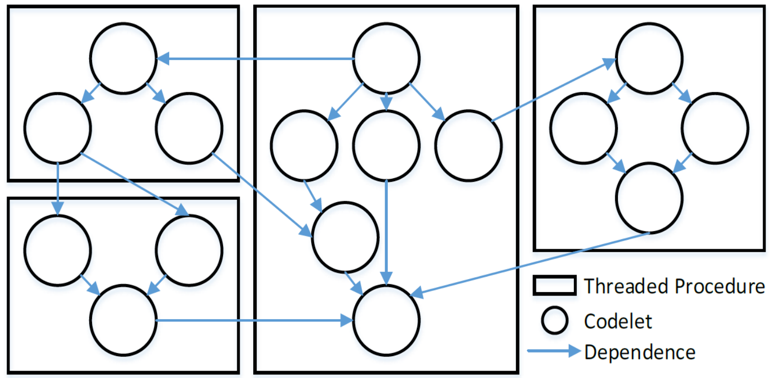
- **<u>Codelet</u>** : A collection of machine instructions which are scheduled atomically as a non-preemptive, single unit of computation
  - Event-driven (**availability of data and resources**)
  - Communicates only through its inputs and outputs
  - Non-preemptive (**cannot be stopped or migrated until end**)



A Codelet

Codelet Model

- **<u>Threaded Procedure</u>** : An asynchronous function which acts as a codelet graph container for a CDG and its needed data
  - Provides a naming convention to invoke a CDG
  - Keep locality of data and computation
  - Associated to a subset of the computational resources

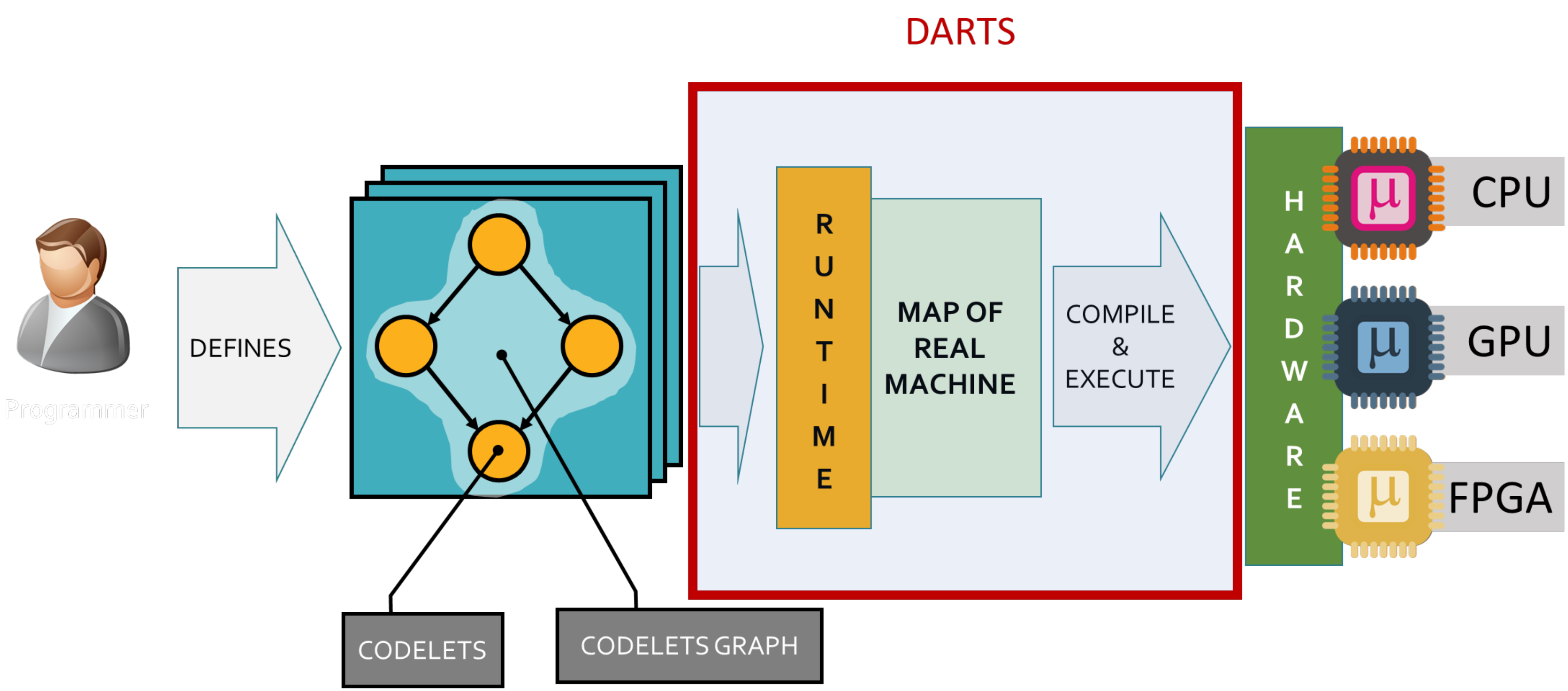


Codelet Graph

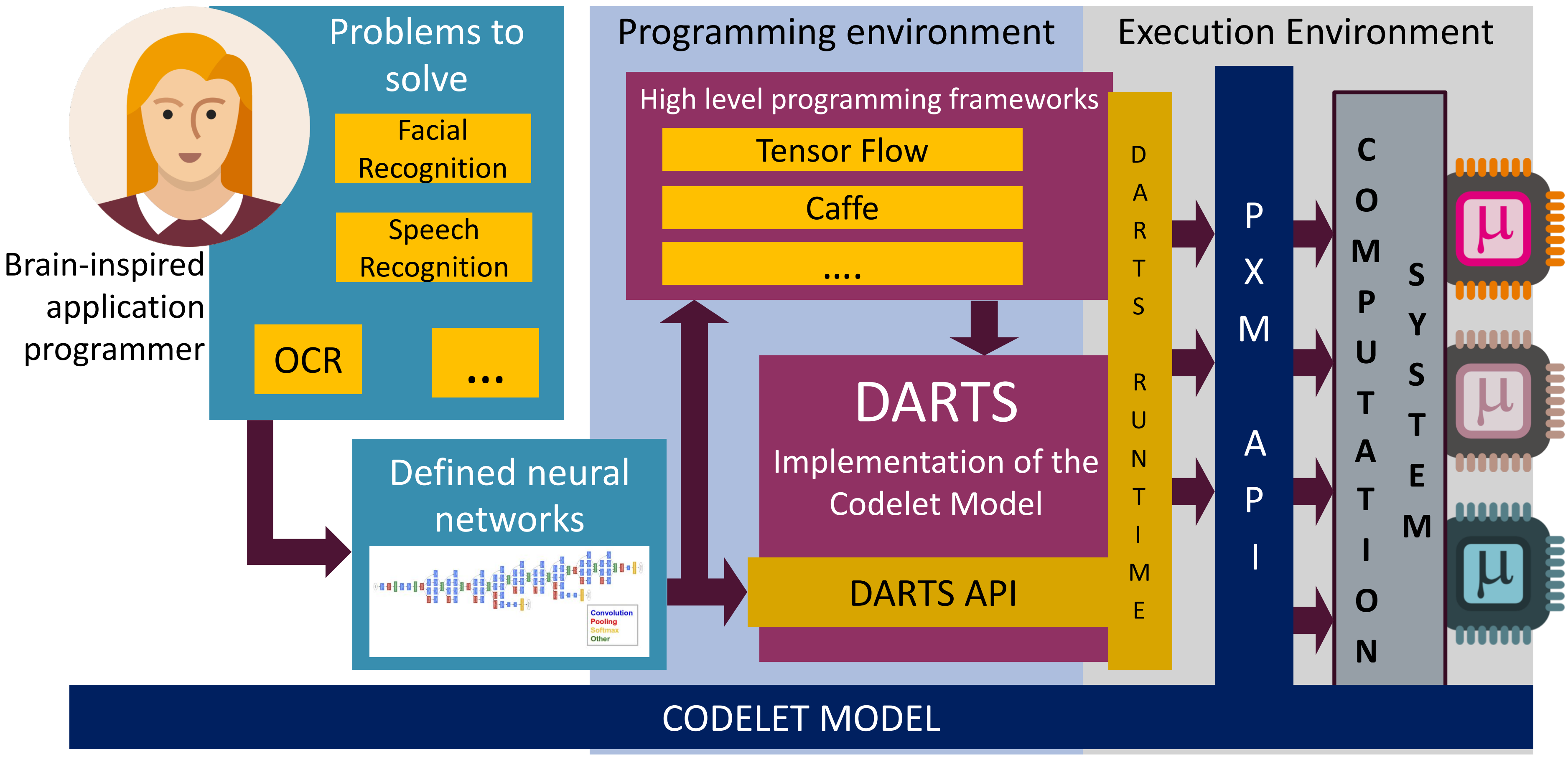
DARTS

- Delaware Adaptive Run-Time System
  - A faithful implementation of Codelet model for single node computing system
- Written in C++
  - Classes are used to represent Codelets and Threaded Procedures
  - Data transmission through shared memory, signal transmission through function calls
- The runtime unburdens the programmer of the execution of the codelets, while providing scheduling policies

System Overview



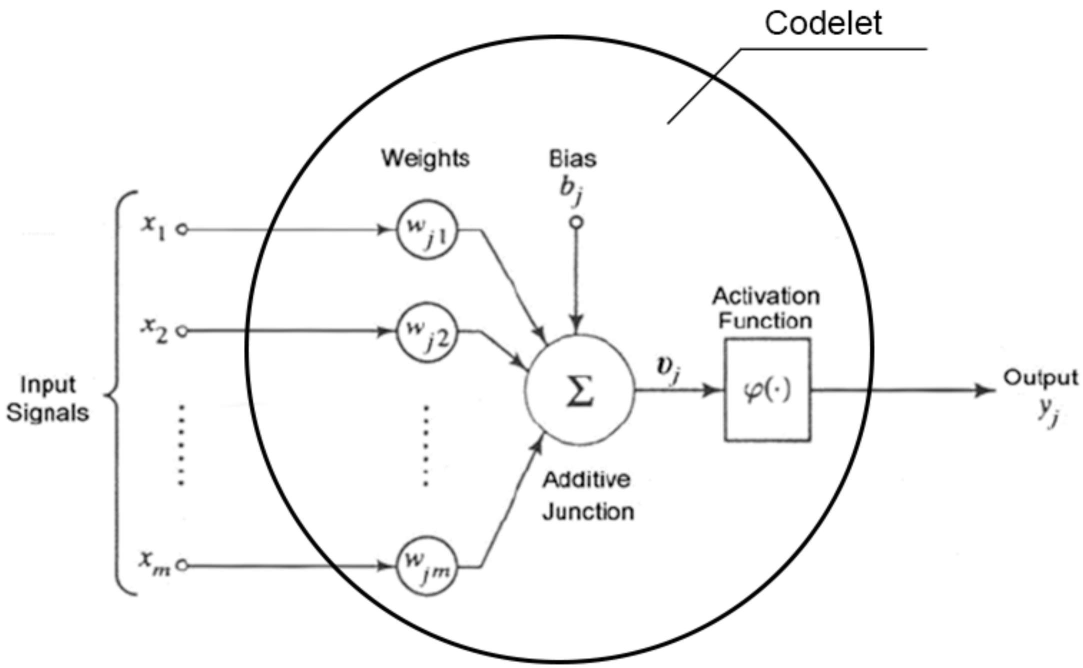
Neural networks



Implementation

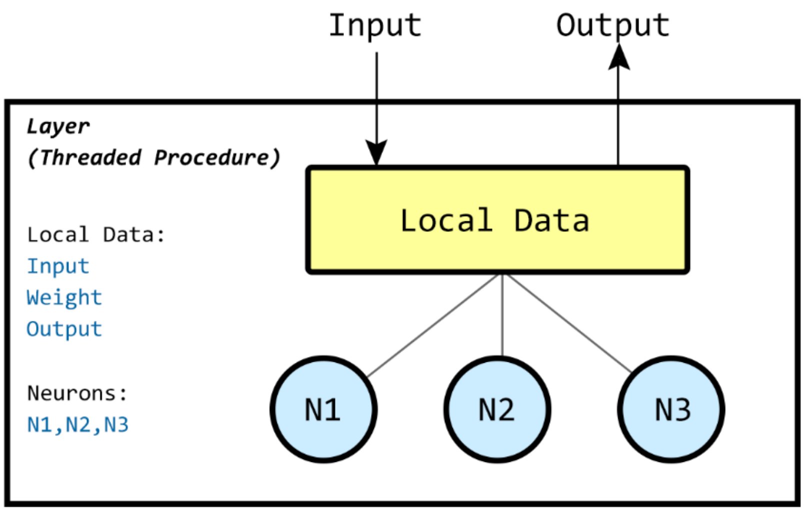
Neuron

- Codelets represent neurons
  - Fire function can be override for different types of neuron.



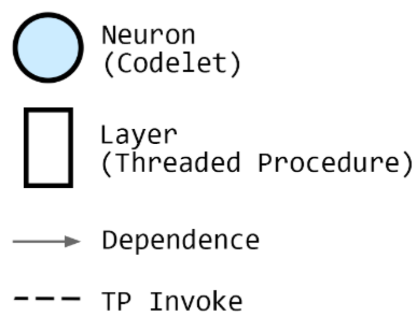
Layer

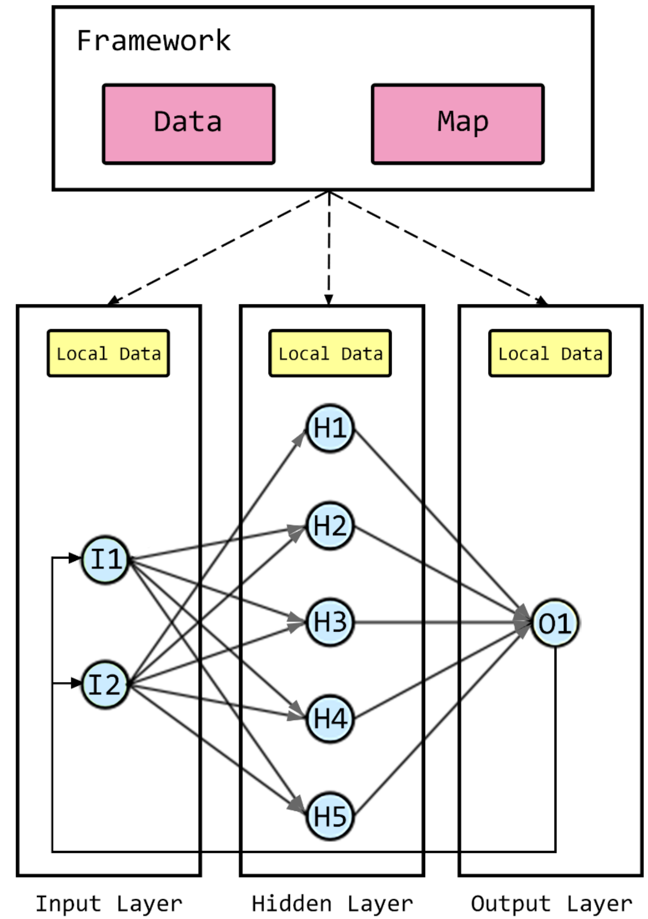
- threaded procedure gather multiple neurons into layers
  - Neurons
  - Local data
- Neurons read from enclosed data in the threaded procedure



Network Building

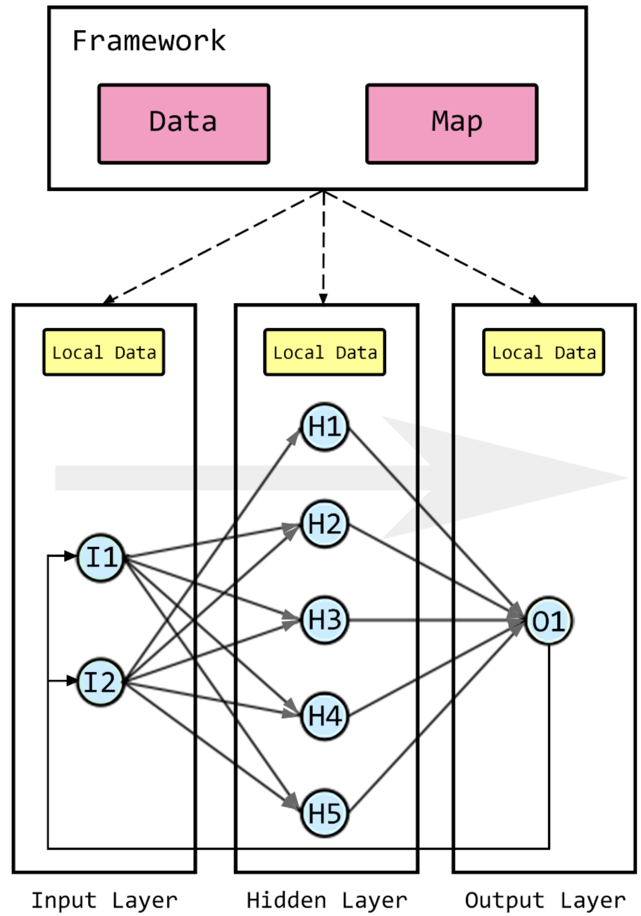
- Multilayer Neural Network
  - Framework to spawn and coordinate layer’s creation
- A framework contains references to each layer:
  - Codelets (neurons)
  - Layer’s Data

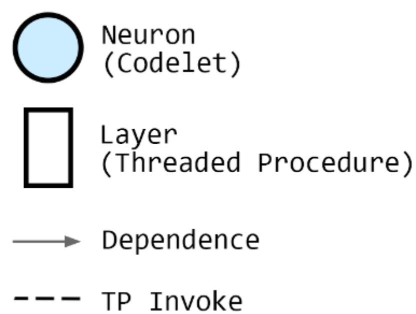




Network Execution

- Asynchrony
  - A neuron will signal its following neuron as long as it finishes without waiting for other neurons in the layer
- Pipelining
  - A neuron will reset itself after firing, preparing itself for the next batch

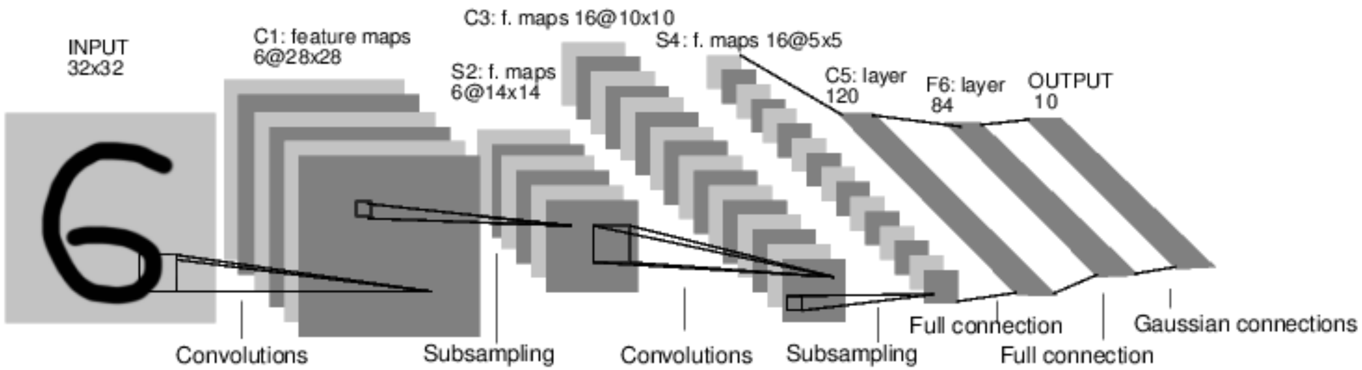




Evaluation

Evaluation

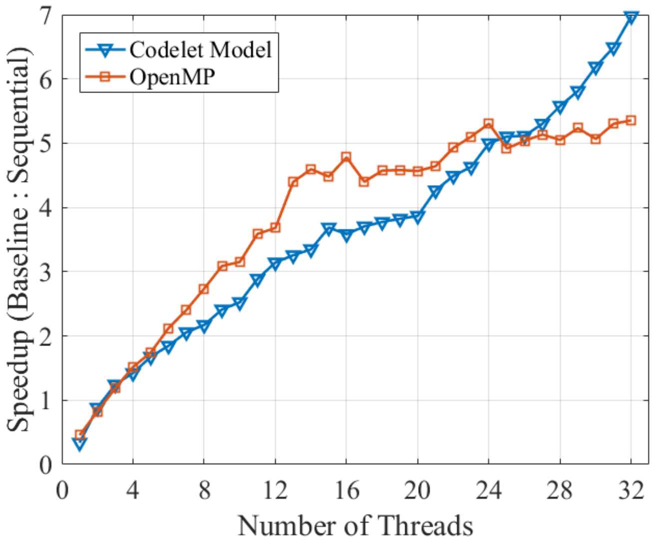
- Main goal
  - Efficiency - Measured by the speedup based on sequential execution.
  - Scalability - Measured by how the performance improves as number of cores increases.
- Simple Benchmark
  - LeNet-5, the most fundamental CNN



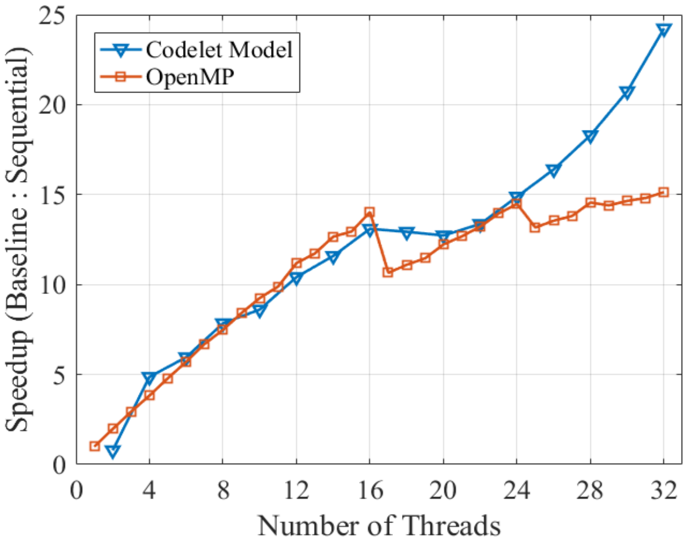
Results

LeNet-5 on MINST (Codelet model VS. OpenMP)

Training

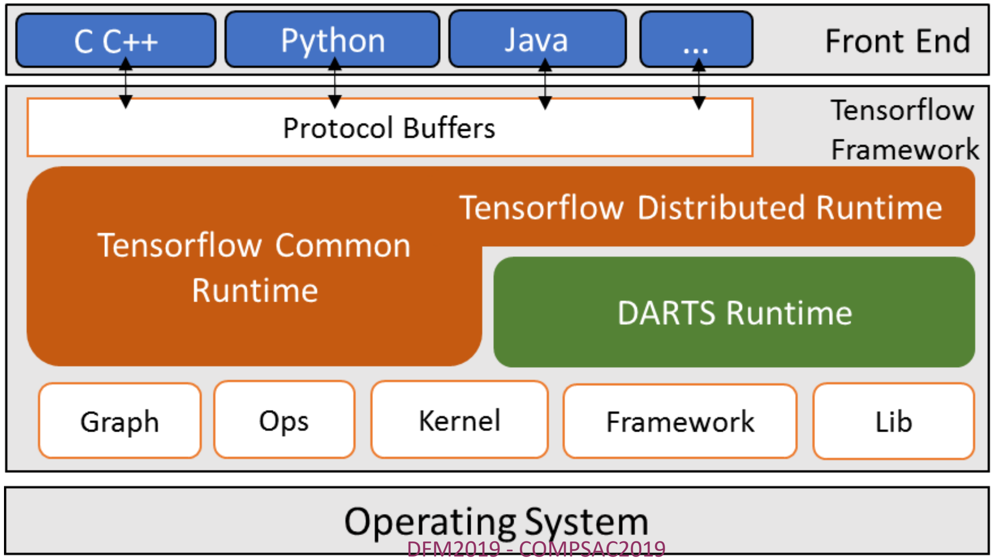


Inference



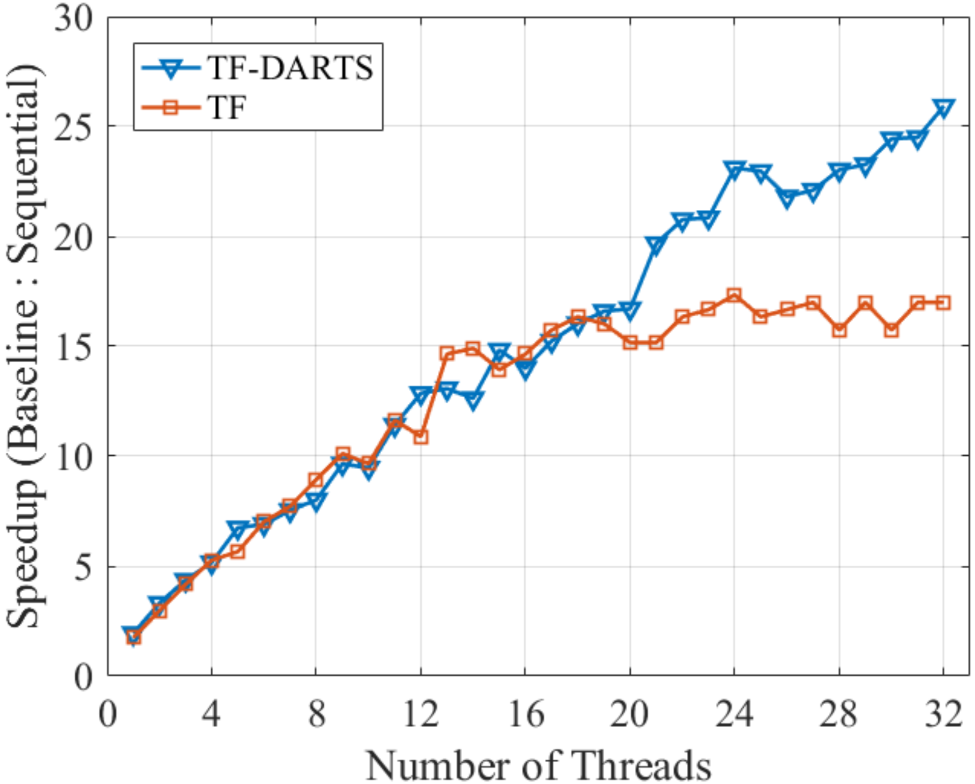
TensorFlow Integration

- Embed Codelet model (called DARTS) into Tensorflow's runtime.
  - DARTS – Single node (Shared memory system)
  - Tensorflow – Multiple nodes (Distributed system)



Results

LeNet-5 on MINST (Integrate DARTS with Tensorflow)



Conclusion

Conclusion

- We develop a fine grain simulation system for brain inspired computing (neural network) on multi-core/heterogeneous system to provide parallel speed up.
- We choose LeNet-5 as benchmark to compare our proposed platform with the model based on OpenMP. The result proves that our platform gives better speedup (up to 62%) compared to OpenMP as number of cores increase. This shows good scalability and high efficiency of codelet based runtime.
- We integrate our platform with tensorflow, opening up a path for deploying our system on distributed system. Result shows that the fusion system outperforms tensorflow alone on a single node when number of threads increases over 16.

Thanks

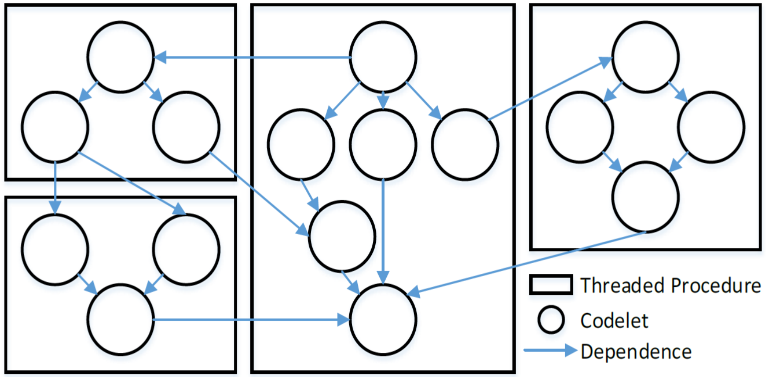
Back up Slides

Innovation

- For the first time combine the Codelet model with brain computing, especially for neural network emulation. Provide a new path for high performance emulation platform for brain computing.
- Adopt fine-grain asynchronous parallel strategy to solve the drawbacks of current parallel strategy, providing a faster way to accelerate the emulation of neural network.
- Fuse our work with Tensorflow, Codelet model doing the computation work and Tensorflow providing the programming interface, make our platform easy to use.

Codelet Model

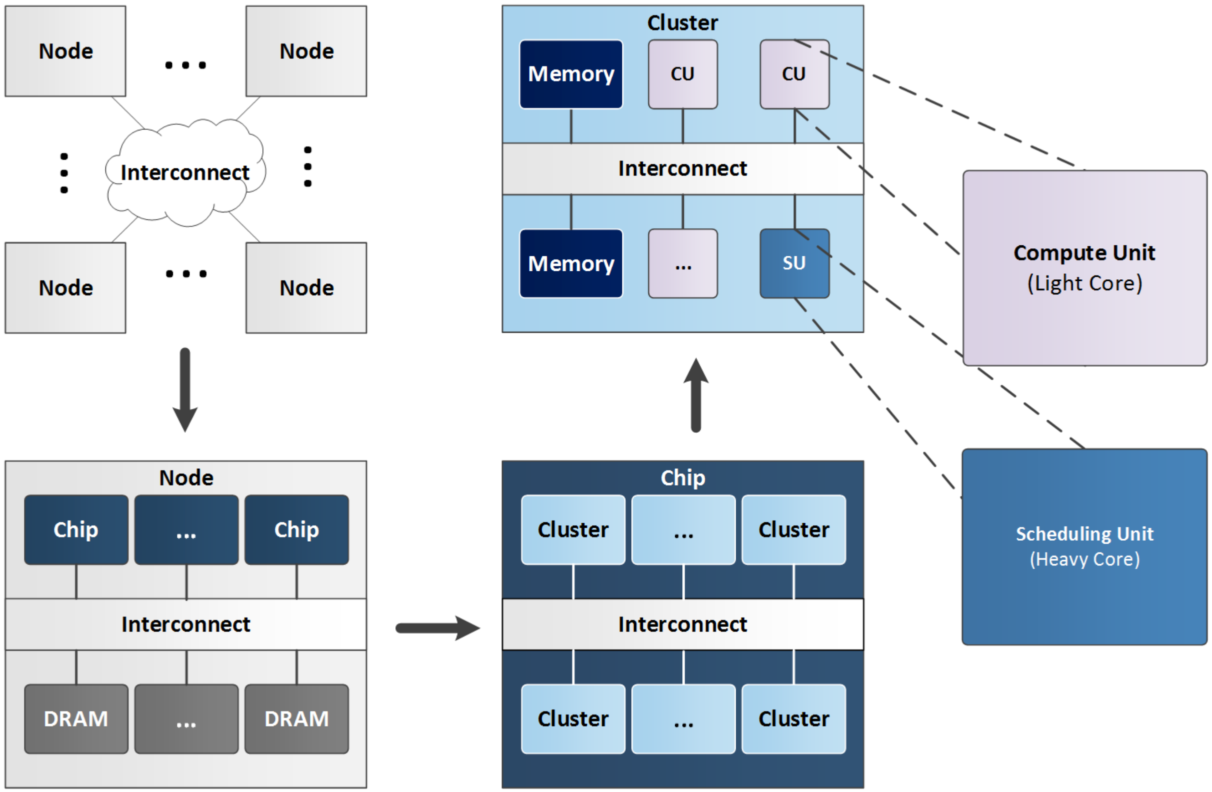
- **Threaded Procedure** : An asynchronous function which acts as a codelet graph container for a CDG and its local data.
  - Provides a naming convention to invoke a CDG.
  - Keep the locality of data. Save time for memory access.



Codelet Graph

Abstract Machine

- Map of the actual architecture.



Abstract Machine

- Cluster
  - A TP is executed by a single cluster in order to load local data into the same cache.
  - TPs are load balance across clusters to make full use of cache.
- Scheduling Unit
  - Load TPs into clusters
  - Distribute Codelet
  - Execute Codelet (When there is no scheduling task)
- Computation Unit
  - Execute Codelet

Future Work

Future Work

- Technical prospective
  - Our platform doesn’t support distributed system, while Tensorflow does. We are trying to further combine them together, to provide a distributed parallel method for brain computing.
- Application prospective
  - Our proposed platform shows high scalability, which will have broad prospects to apply many-core chips to brain computing emulation.
  - Codelet model is event-driven, the events also include the requirement of conditions such as power. We hope this will help to develop the ultra-low power devices.