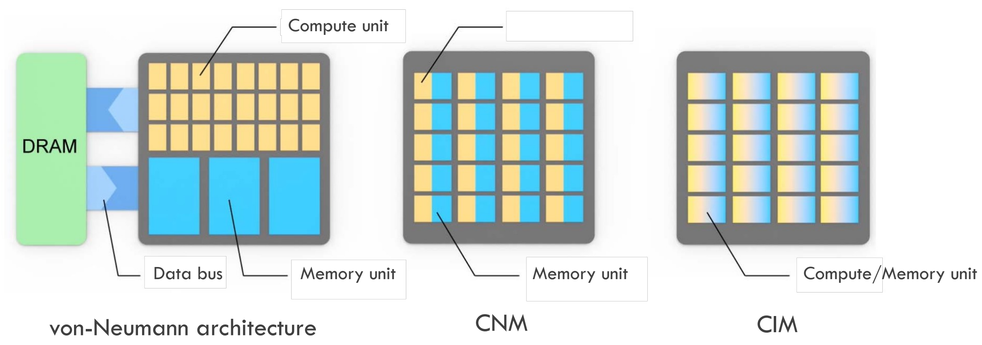
HetCIM: Compilation for heterogenous compute-in/near-memory systems

Asif Ali Khan, Hamid Farzaneh, João Paulo Cardoso de Lima, Jeronimo Castrillon
Contact: asif_ali.khan@tu-dresden.de

Motivation

- Data movement accounts for more than 60% of total system energy[1]
- Compute-in-memory (CIM) and compute-near-memory (CNM) systems allow computations in the data proximity



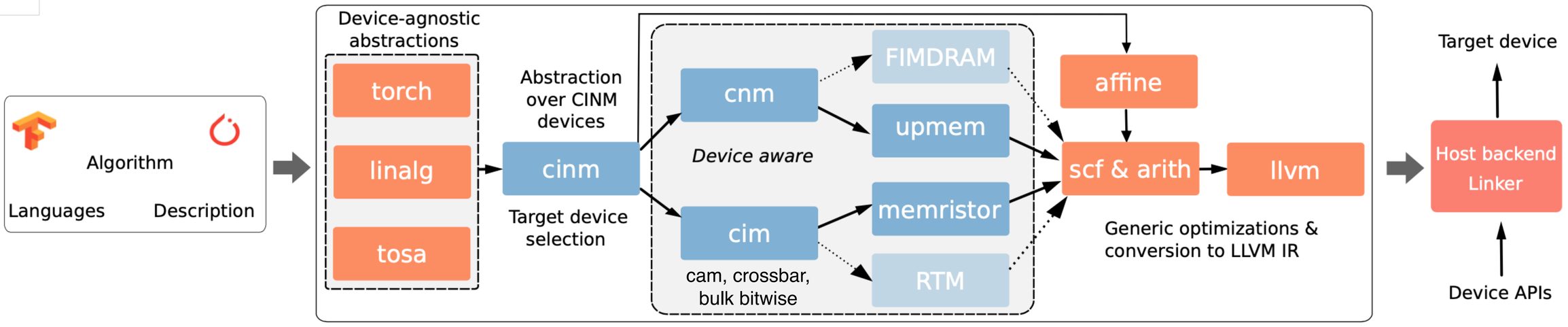
- Most of these systems provide low-level device libraries

```
...
uint32_t mram_base_addr_C = (uint32_t) (DPU_MRAM_HEAP_POINTER + 2 * ROWS * COLS * sizeof(T));
for(int i = (tasklet_id * point_per_tasklet) ; i < ( (tasklet_id+1)*point_per_tasklet ) ;
↪  i++) {
    if( new_row != row ){
        ...
        mram_read((__mram_ptr void const*) (mram_base_addr_A + mram_offset_A), cache_A, COLS
        ↪  * sizeof(T));
    }
    mram_read((__mram_ptr void const*) (mram_base_addr_B + mram_offset_B), cache_B, COLS *
    ↪  sizeof(T));
    dot_product(cache_C, cache_A, cache_B, number_of_dot_products);
    ...
}
...
```

- Programmability of these systems is a **serious challenge** as future systems are predicted to be heterogenous

Software stack for heterogeneous CIM and CNM systems

- A hierarchal compilation framework (Cinnamon) based on MLIR that abstracts over CIM and CNM devices
- Implements hierarchal device-(in)dependent analysis and transformations
- Maps computational patterns to most suitable backend targets
- **Input:** High-level program representation (TensorFlow, PyTorch, etc)
- **Targets:** UPMEM, crossbars, CAMs, bulk bitwise logic



Use cases: Different backend targets and optimizations

Compilation for memrisitive crossbars

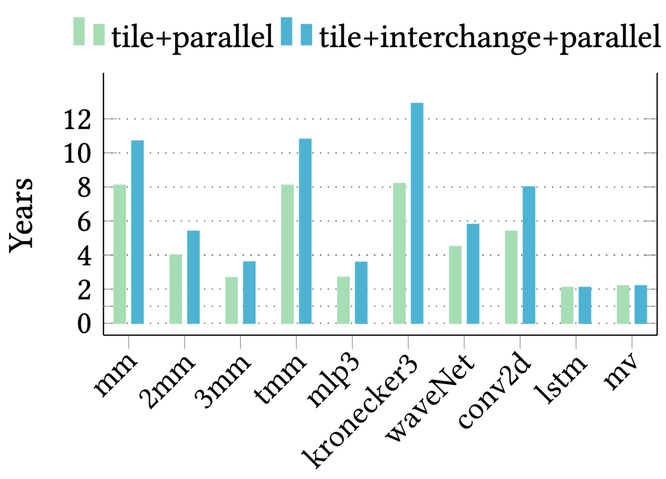
- Analog multiplication with 4 tiles each 64x64
- Maximize parallelism and minimize writes
- Target(s): Lifetime, energy and performance

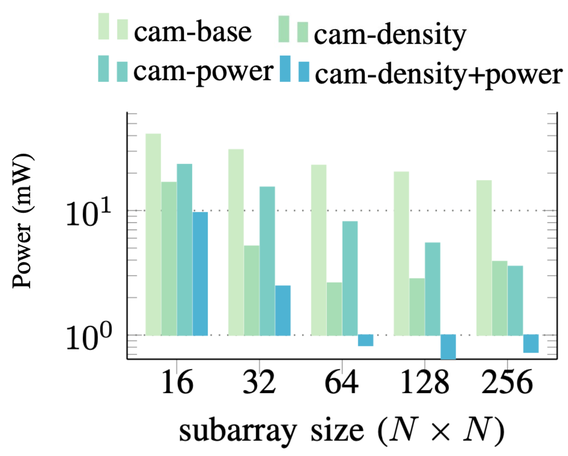
Compilation for CAM

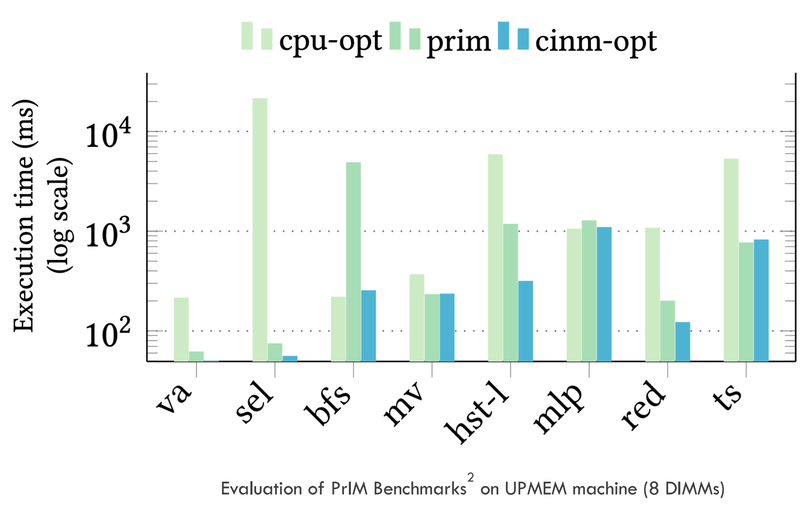
- Parallel search operations with FeFET- CAMs
- Compulsory partitioning: fit kernels onto CAMs
- Target(s): Density and power optimization

Compilation for CNM[2]

- General-purpose UPMEM system
- Load balancing and locality improvement
- Target: Minimize execution time







Evaluation of PrIM Benchmarks[2] on UPMEM machine (8 DIMMs)

1. A. Boroumand, S. Ghose, Y. Kim, R. Ausavarungnirun, E. Shiu, R. Thakur, D. Kim, A. Kuusela, A. Knies, P. Ranganathan, O. Mutlu, Google Workloads for Consumer Devices: Mitigating Data Movement Bottlenecks, in: ASPLOS, 2018.
2. Gómez-Luna et al. "Benchmarking a New Paradigm: Experimental Analysis and Characterization of a Real Processing-in-Memory System," in IEEE Access, vol. 10, pp. 52565-52608, 2022, doi: 10.1109/ACCESS.2022.3174101.
3. Siemieniuk, L. Chelini, A. A. Khan, J. Castrillon, A. Drebes, H. Corporaal, T. Grosser, M. Kong, "OCC: An Automated End-to-End Machine Learning Optimizing Compiler for Computing-In-Memory", In IEEE TCAD, Jul 2021

cfaed.tu-dresden.de