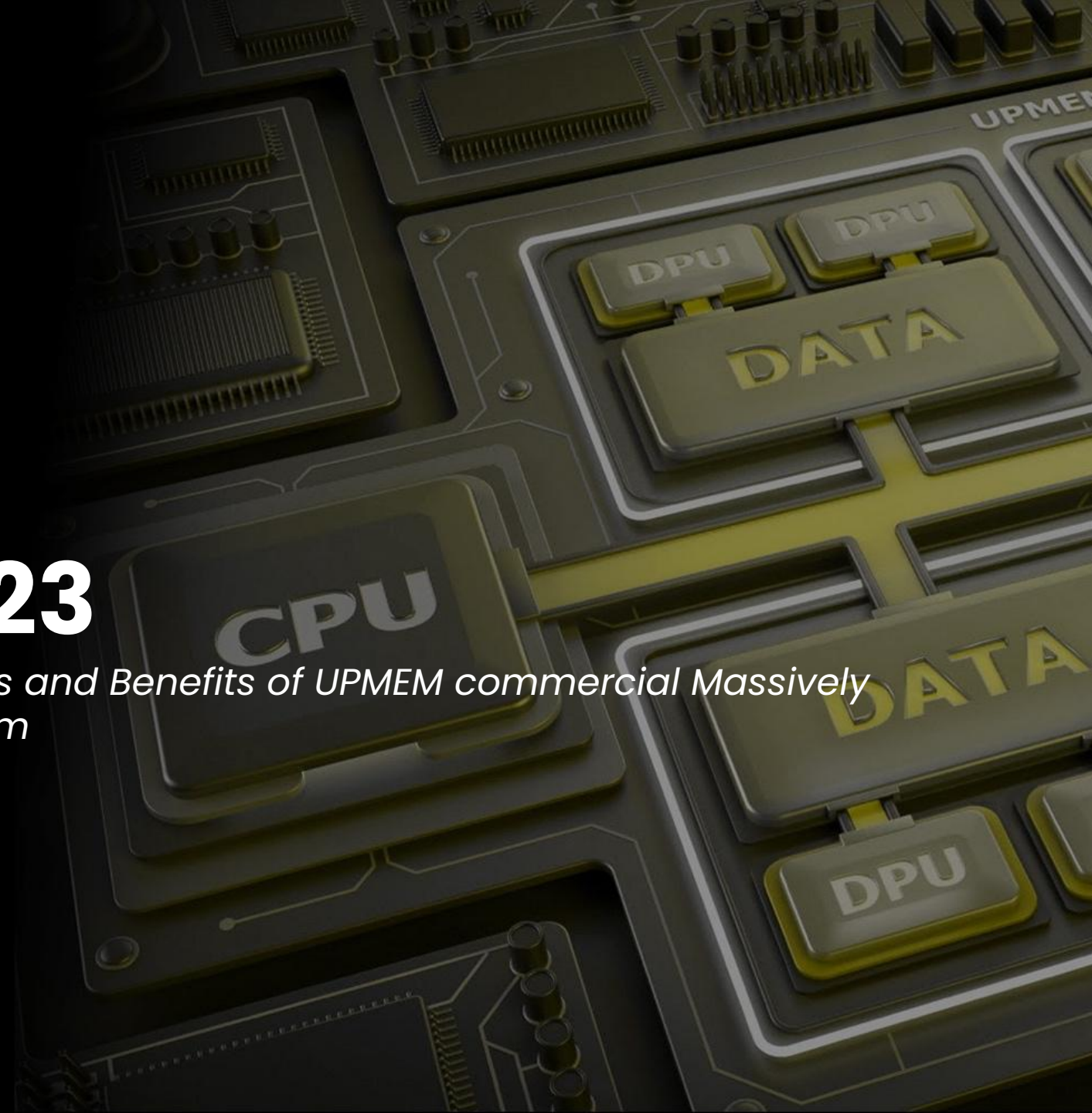


ABUMPIMP 2023

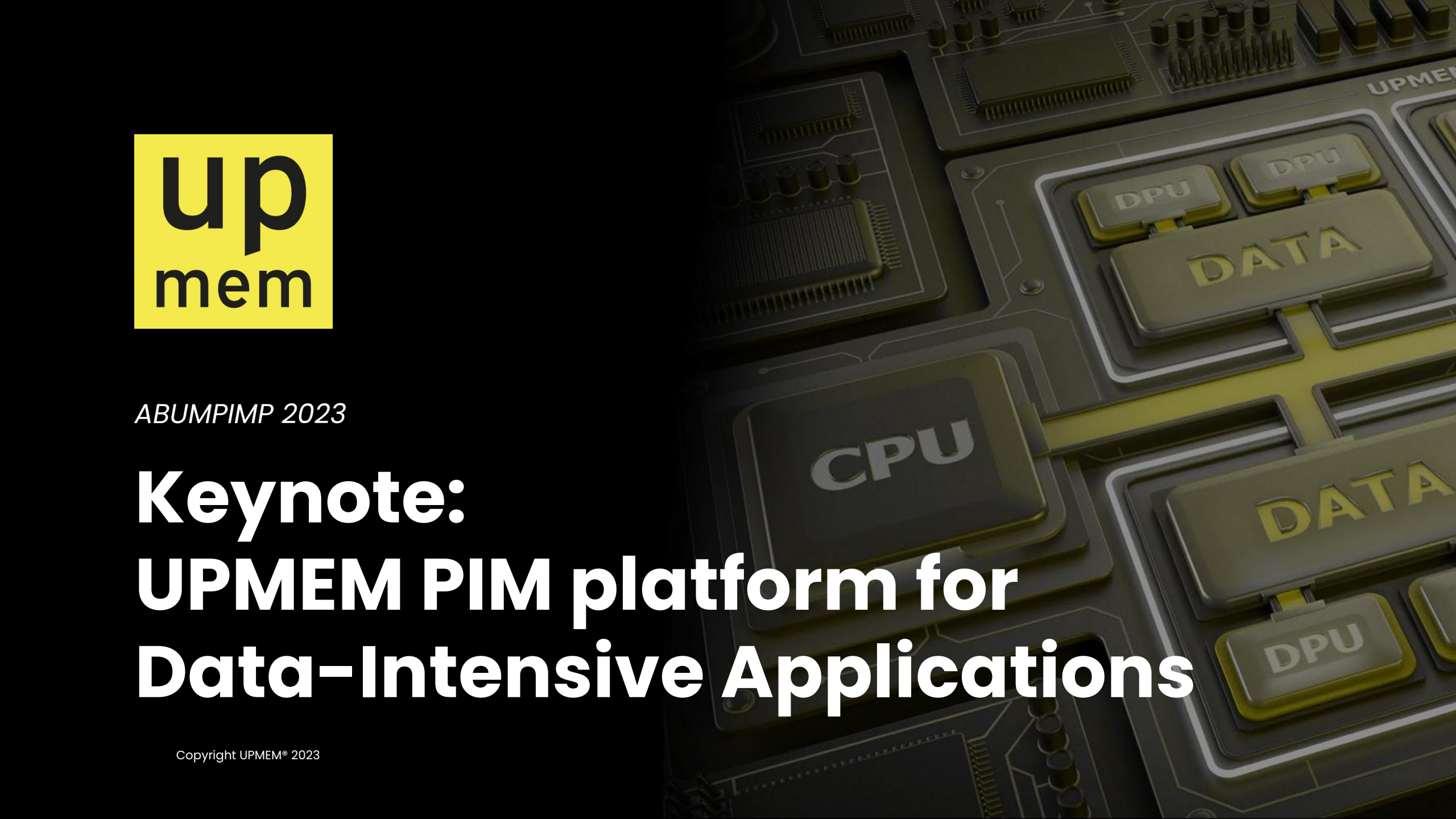
The 1st Minisymposium on Applications and Benefits of UPMEM commercial Massively Parallel Processing-In-Memory Platform

August 29, 2023

EURO-PAR CONFERENCE 2023

Today's agenda

TIME	TITLE	SPEAKER(S)
09:00 – 09:05	Session welcome and aims	UPMEM
09:05 – 10:00	Keynote: UPMEM PIM platform for Data-Intensive Applications	Yann FALEVOZ (UPMEM) / Julien LEGRIEL (UPMEM)
10:00 – 10:30	Invited talk: Understanding the potential of real processing-in-memory for modern workloads	Juan GOMEZ LUNA (ETHZ)
10:30 – 11:00	Coffee break	—
11:00 – 11:30	Research paper: pimDB: From Main-Memory DBMS to Processing-In-Memory DBMS-Engines on Intelligent Memories	Arthur BERNHARDT (Reutlingen University)
11:30 – 12:00	Invited talk: A Fast Processing-in-DIMM Join Algorithm Exploiting UPMEM DIMMs	Chaemin LIM (Yonsei University)
12:00 – 12:30	Invited talk: PIM Performance and Economics for In-Memory Databases	Hanna KRUPPE (SAP)
12:30 – 14:00	Lunch Break	—
14:00 – 14:30	Research paper: Banded Dynamic Programming Algorithms on UPMEM PIM Architecture	Meven MOGNOL (Univ. Rennes, CNRS-IRISA, Inria & UPMEM)
14:30 – 15:00	Research paper: Protein Alignment on UPMEM PIM Architecture	Dominique LAVENIER (Univ. Rennes, CNRS-IRISA & Inria)
15:00 – 15:30	Research paper: An Experimental Evaluation of Machine Learning Training on a Real Processing-in-Memory System	Juan GOMEZ LUNA (ETHZ) / Sylvan BROCARD (UPMEM)
15:30 – 16:00	Coffee break	—
16:00 – 16:30	Research paper: Implementation and Evaluation of Deep Neural Networks in Commercially Available Processing in Memory Hardware	Purab SUTRADHAR (RIT)
16:30 – 17:00	Research paper: Privacy-Preserving Computing on UPMEM	Elaheh SADREDINI (UCR)
17:00 – 17:15	Closing	UPMEM



ABUMPIMP 2023

Keynote: UPMEM PIM platform for Data-Intensive Applications

Keynote's agenda

1. Technology Overview
2. High-Level Hardware Architecture
3. Programming the UPMEM PIM
4. PIM Applications
5. Hardware Roadmap

Overcome data and energy bottleneck thanks to PIM

Founded: 2015

Employee Count: ~20

Headquarters: Grenoble, France

Total Patents: 11

Gilles Hamou

CEO / Co-Founder

Track Record:

Co-owner @ Oscaro.com

Scaled Oscaro.com to $100M revenue from inception

Founded & scaled Plantes-et-Jardins.com

Senior Manager @ RSM

Case Leader @ BCG

Education:

MBA INSEAD

Eng. Centrale Paris

Fabrice Devaux

CTO / Co-Founder

Track Record:

Senior Staff SWE @ VMWare

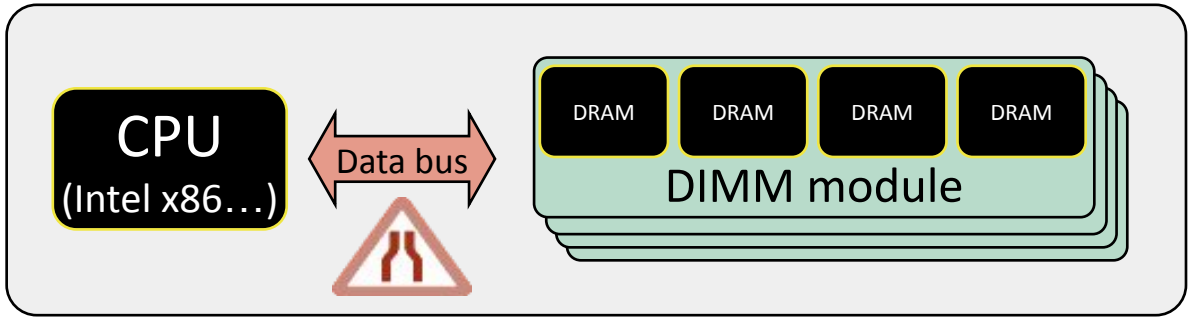
Co-owner, CTO @ Trango Virtual Processors, sold to VMware

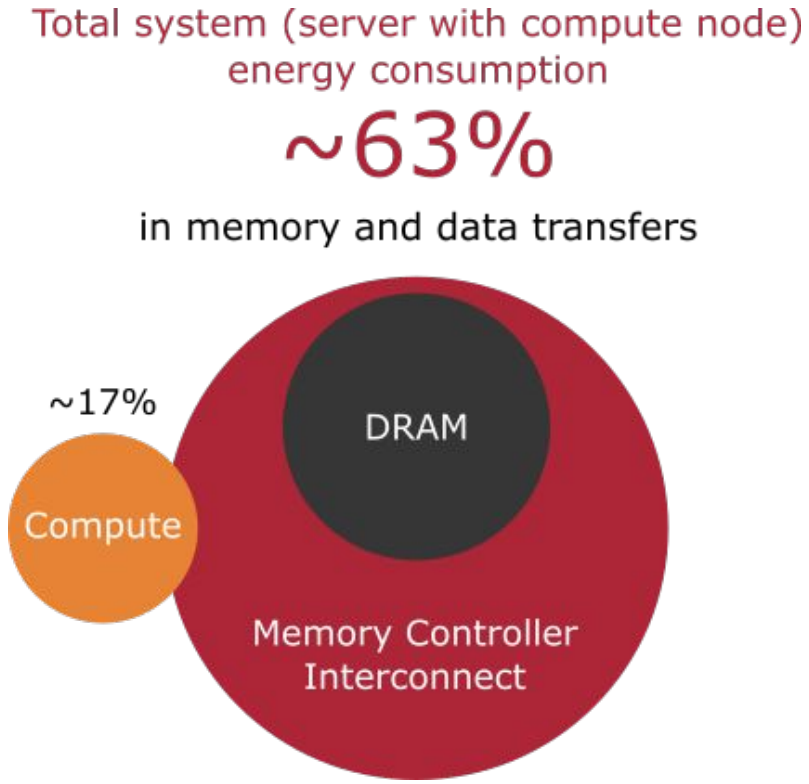
CPU Architect @ STMicroeletronics

Education:

DEA, Microelectronics, Pierre and Marie Curie University

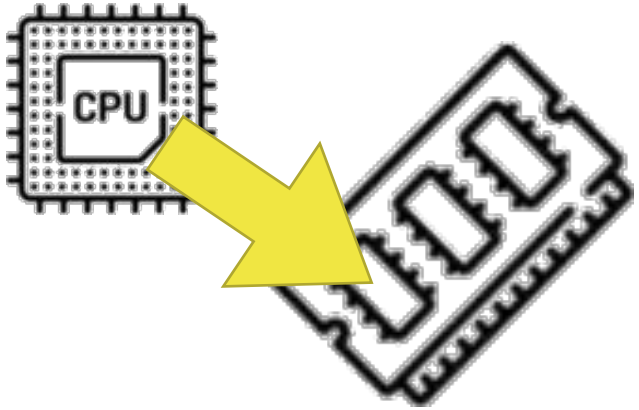
Limitations of Traditional Compute-Centric Architectures for Data-Intensive Workloads

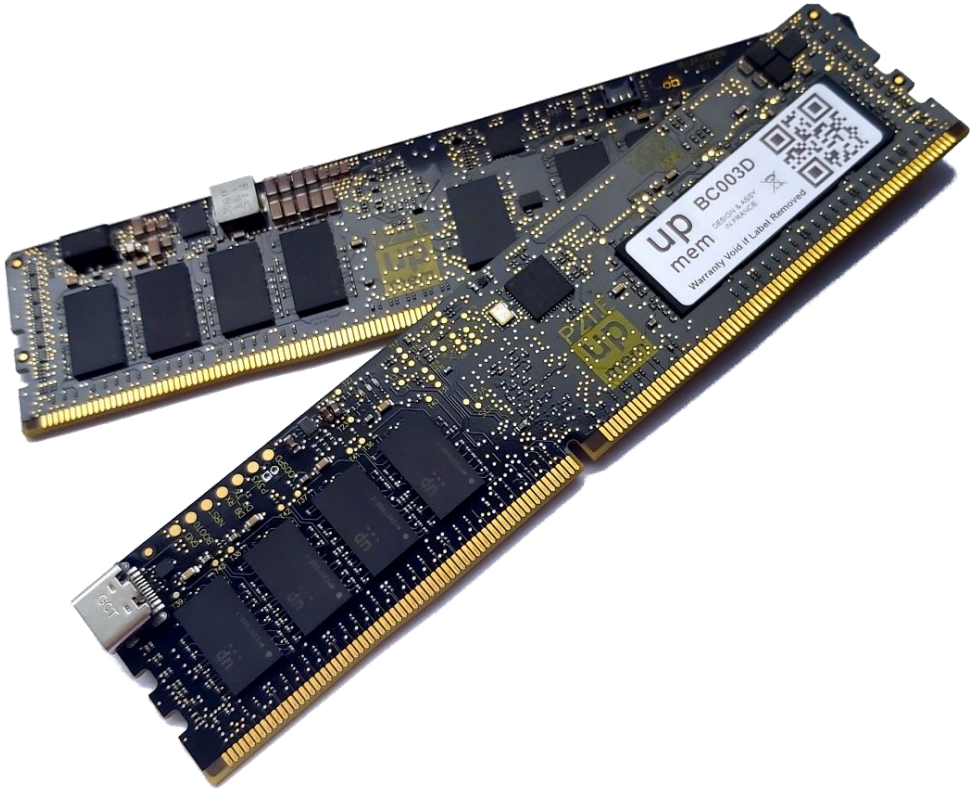




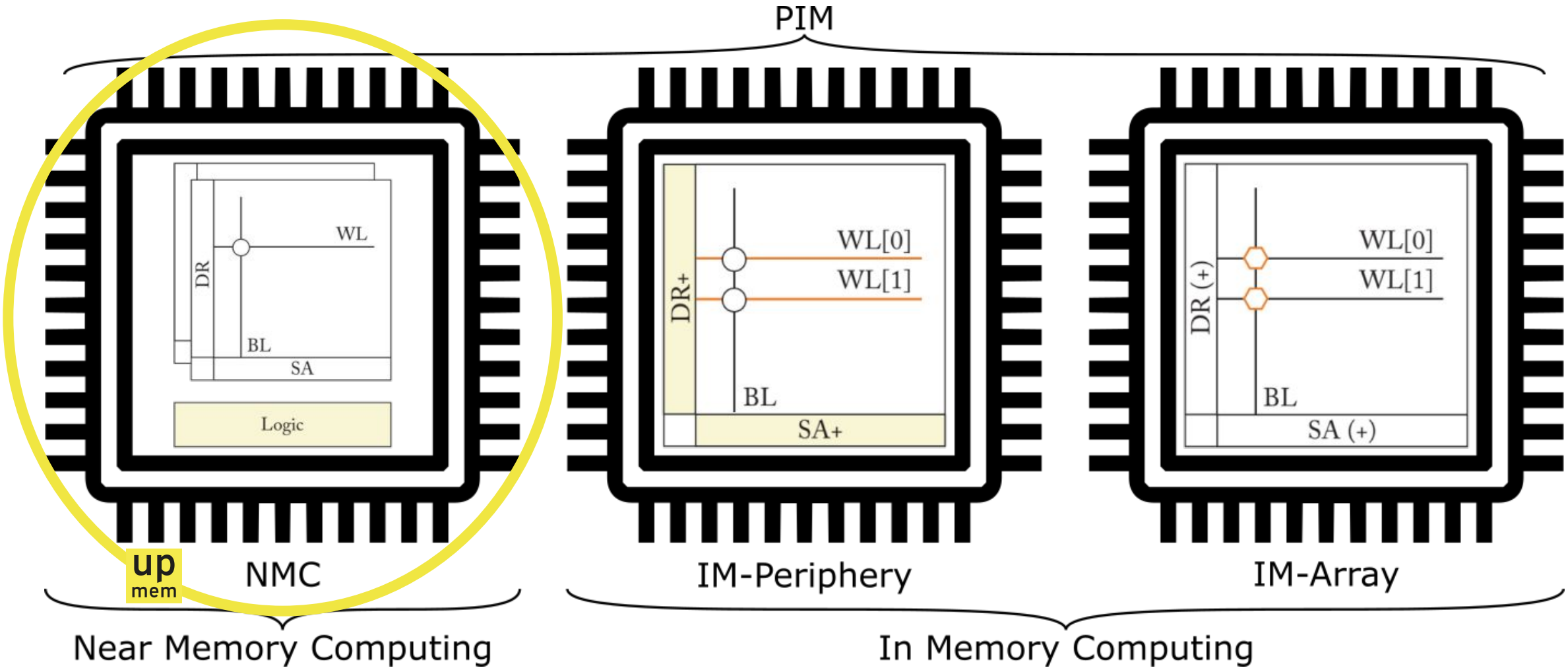
Source: SK hynix CEO Seok-Hee Lee's keynote at the GSA Memory+ 2021 conference, confirming Lawrence Berkeley Lab results.

Overcome data and energy bottleneck thanks to processing in memory





Taxonomy of processing in memory (PIM)



Source: Das, Reetuparna., Wang, Xiaowei., Fujiki, Daichi., Subramaniyan, Arun. In-/near-Memory Computing. United States: Morgan & Claypool Publishers, 2021.

Proven capacity to benefit a wide range of applications

Genomics

Databases

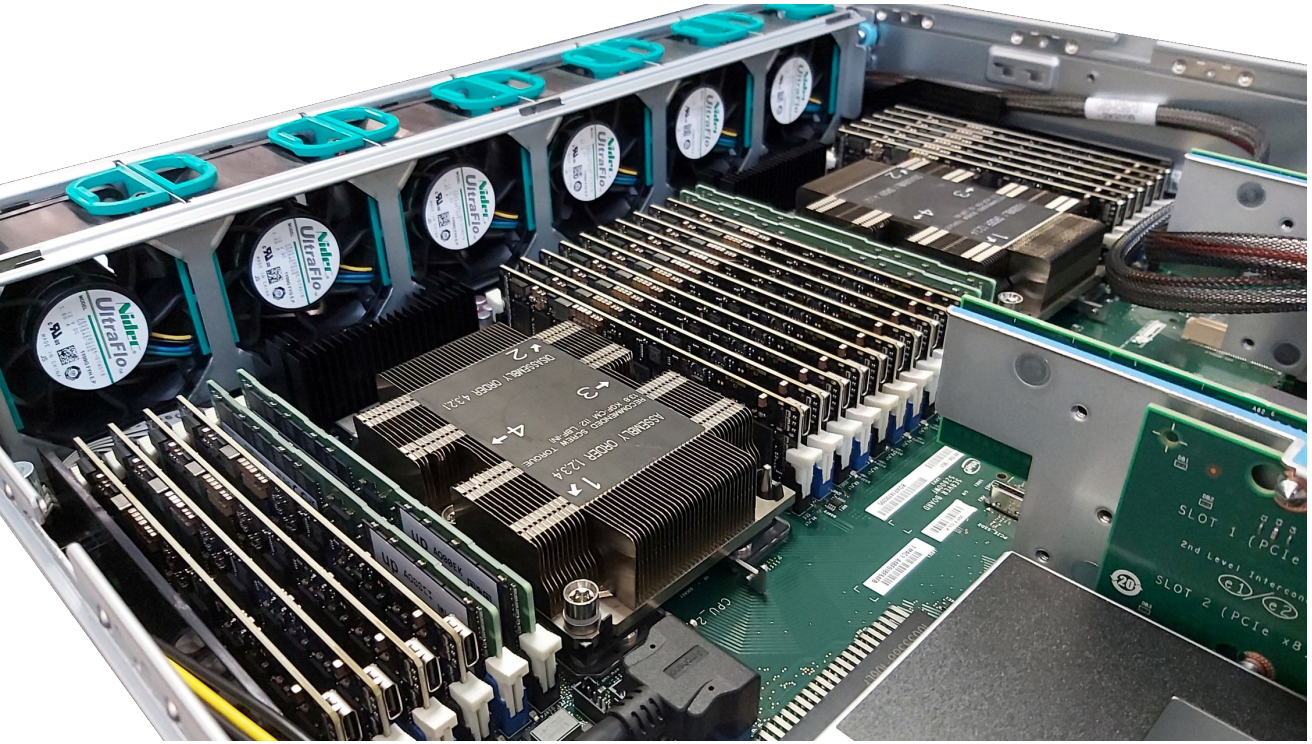
Data analytics

Image Processing

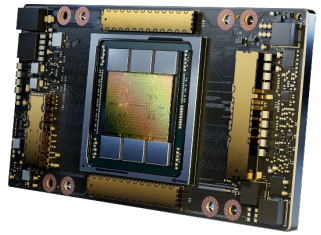
AI

Cryptography

Compression



VS

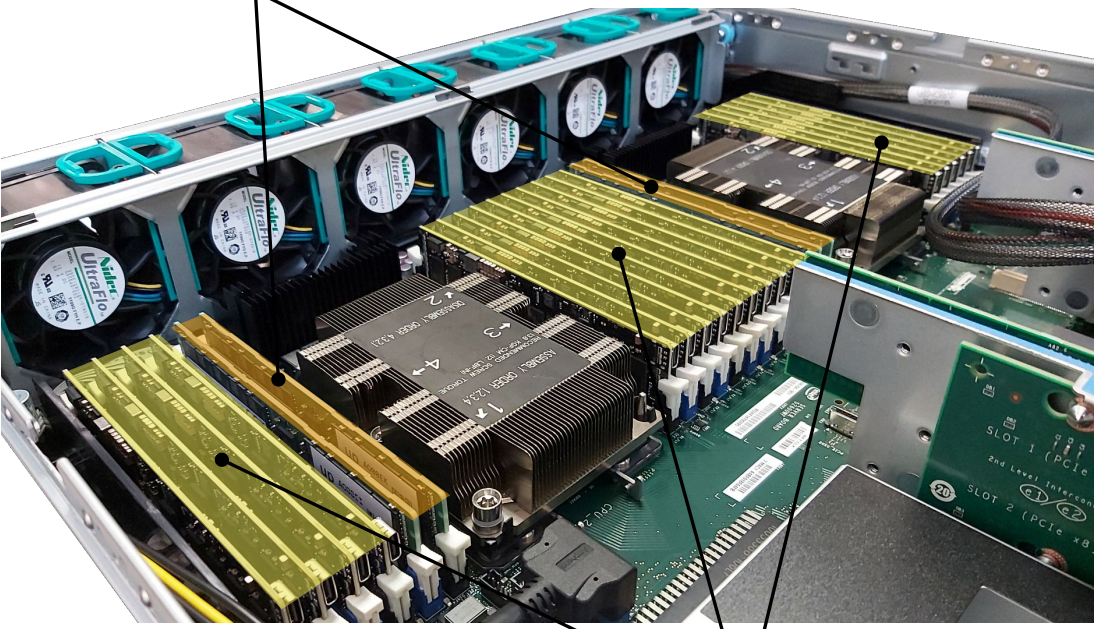


up mem

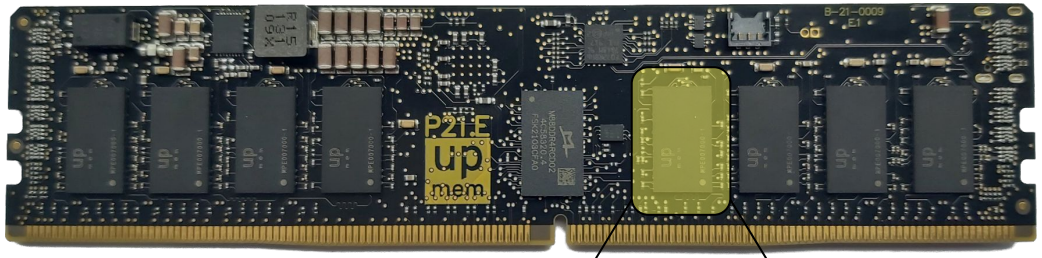
Technology Overview

A standard application server populated with PIM DIMMs

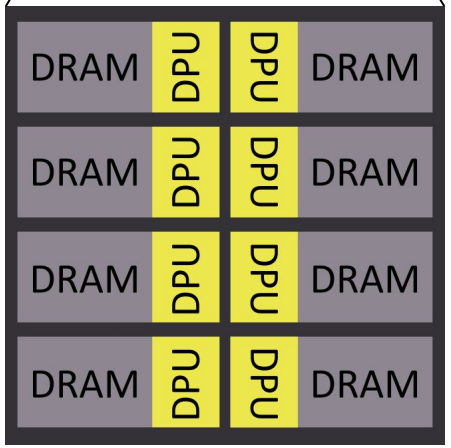
4 Standard DIMMS



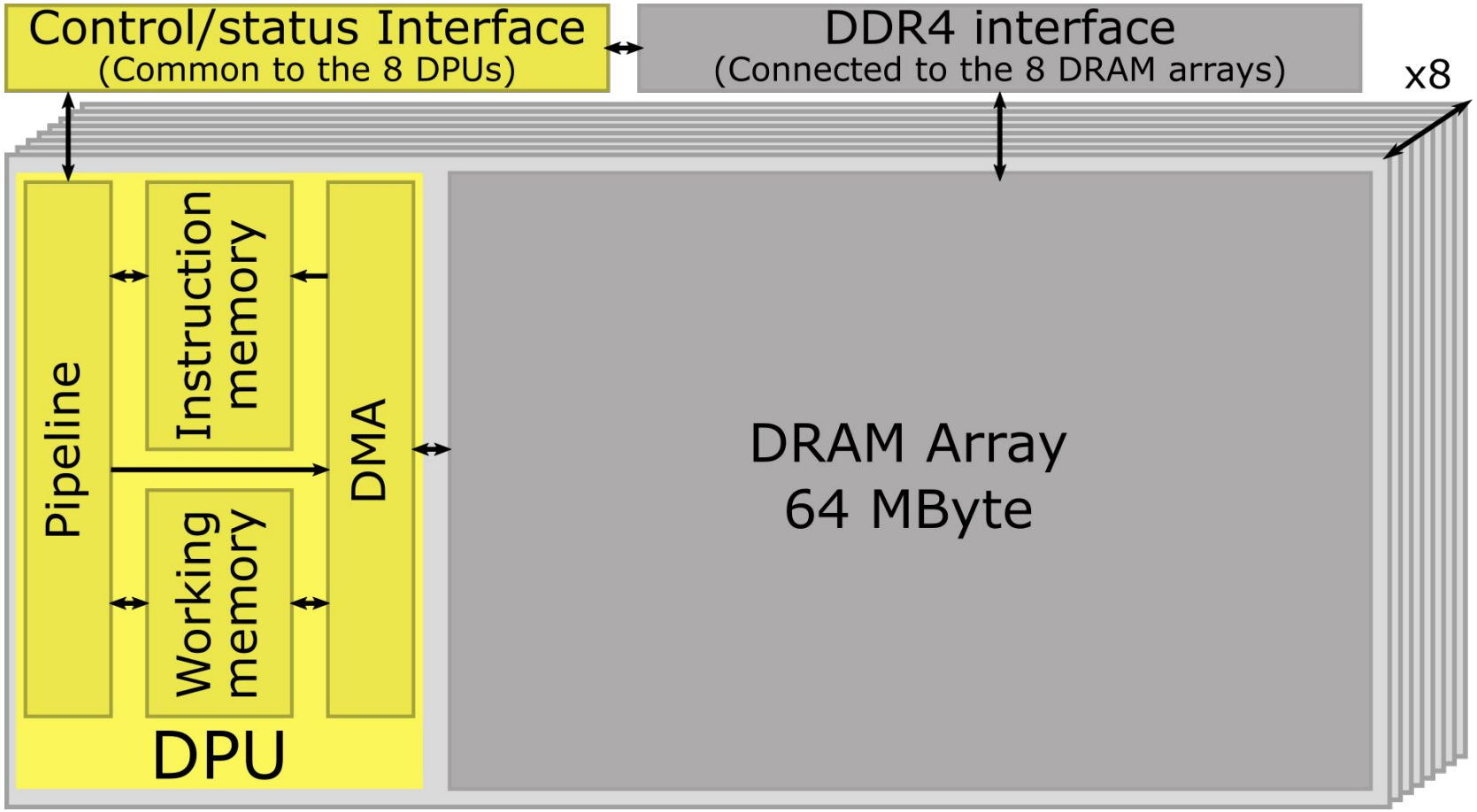
20 PIM DIMMS



2560 DPUs for 160GB of PIM DRAM



A DPU is a simple modern general-purpose processor



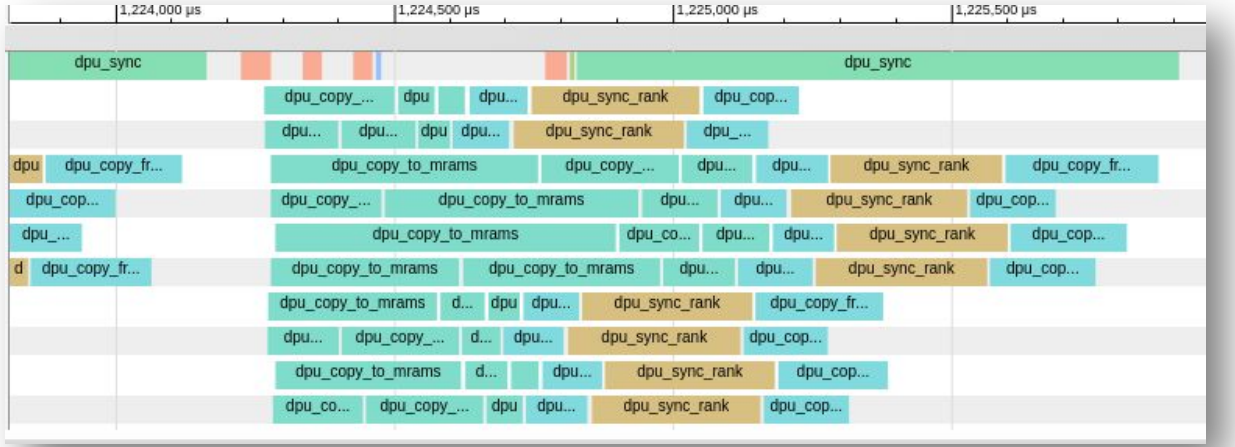
- Shared access with the host CPU to a DRAM bank
- Instruction and data caches replaced by instruction RAM and a Working RAM.
- Independent and asynchronous
- 24 independent threads per DPU
- No direct communication channel among DPUs

A set of tools for smooth application porting

x86 program written in C, C++ or python with C functions to call routines on the DPUs

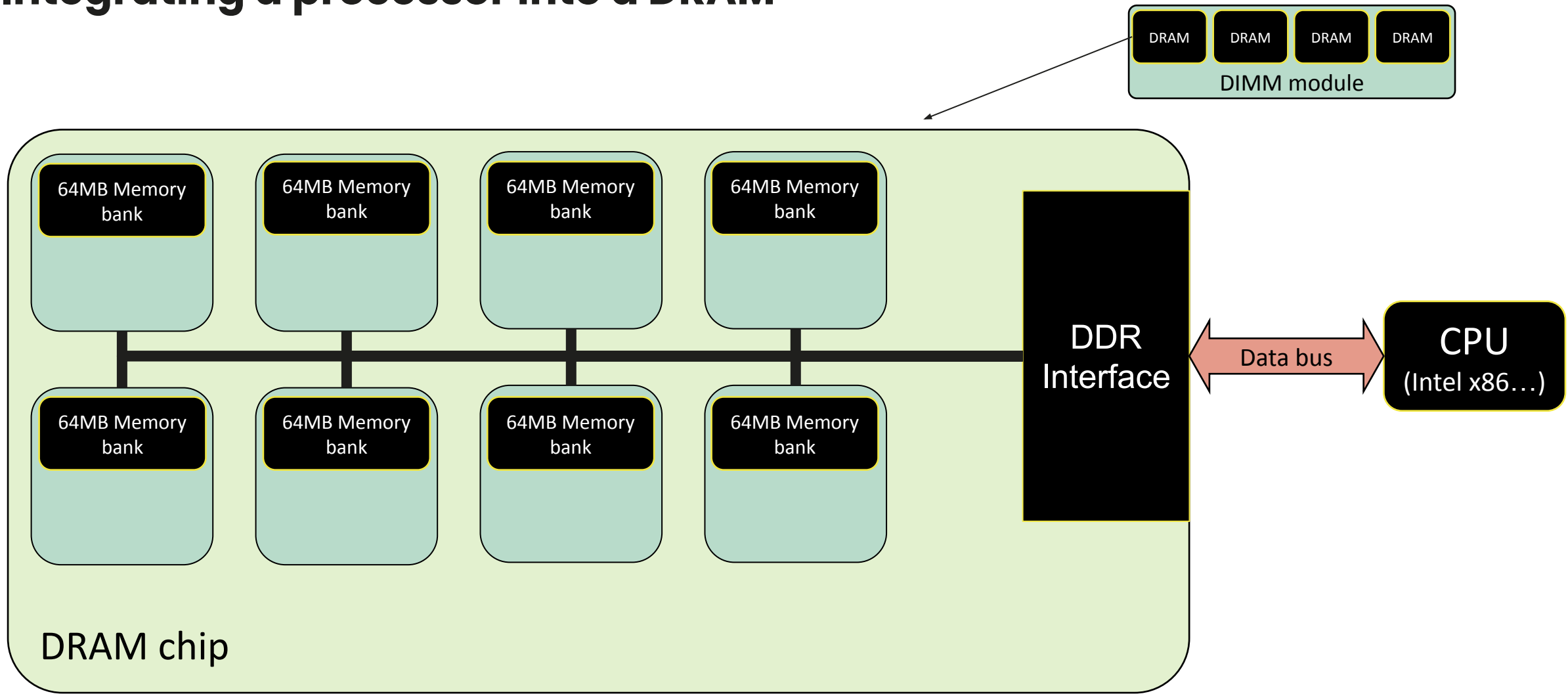
UPMEM SDK contains:

- A Full-featured runtime library for the DPU
- Management and communication libraries to encapsulate easily all the Host to DPU operations
- An LLVM based C-compiler using LLVM 12.0
- A LLDB based debugger
- Programming tools: profilers, simulator…
- Server BIOS binaries
- Linux driver for x86 servers Validated on Redhat, Ubuntu and Debian.

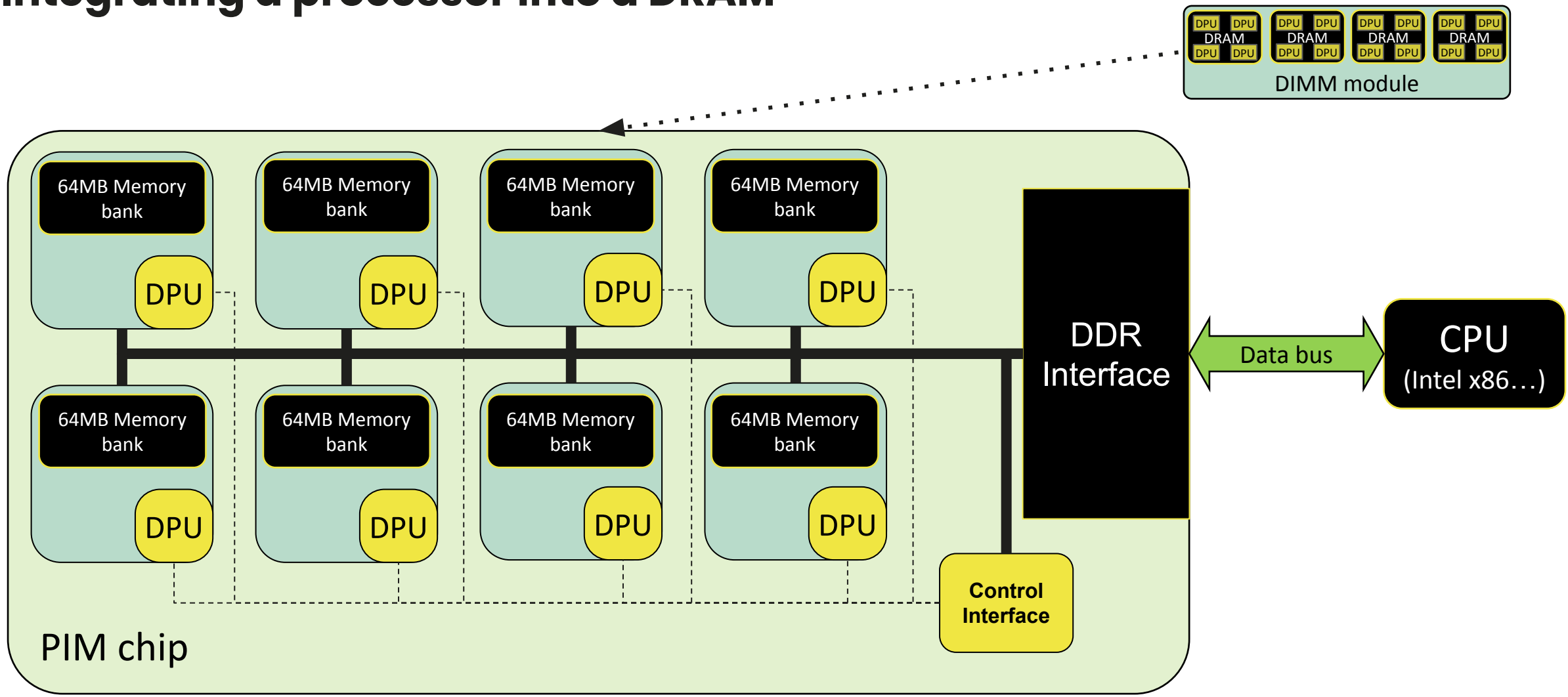


High Level Hardware Architecture

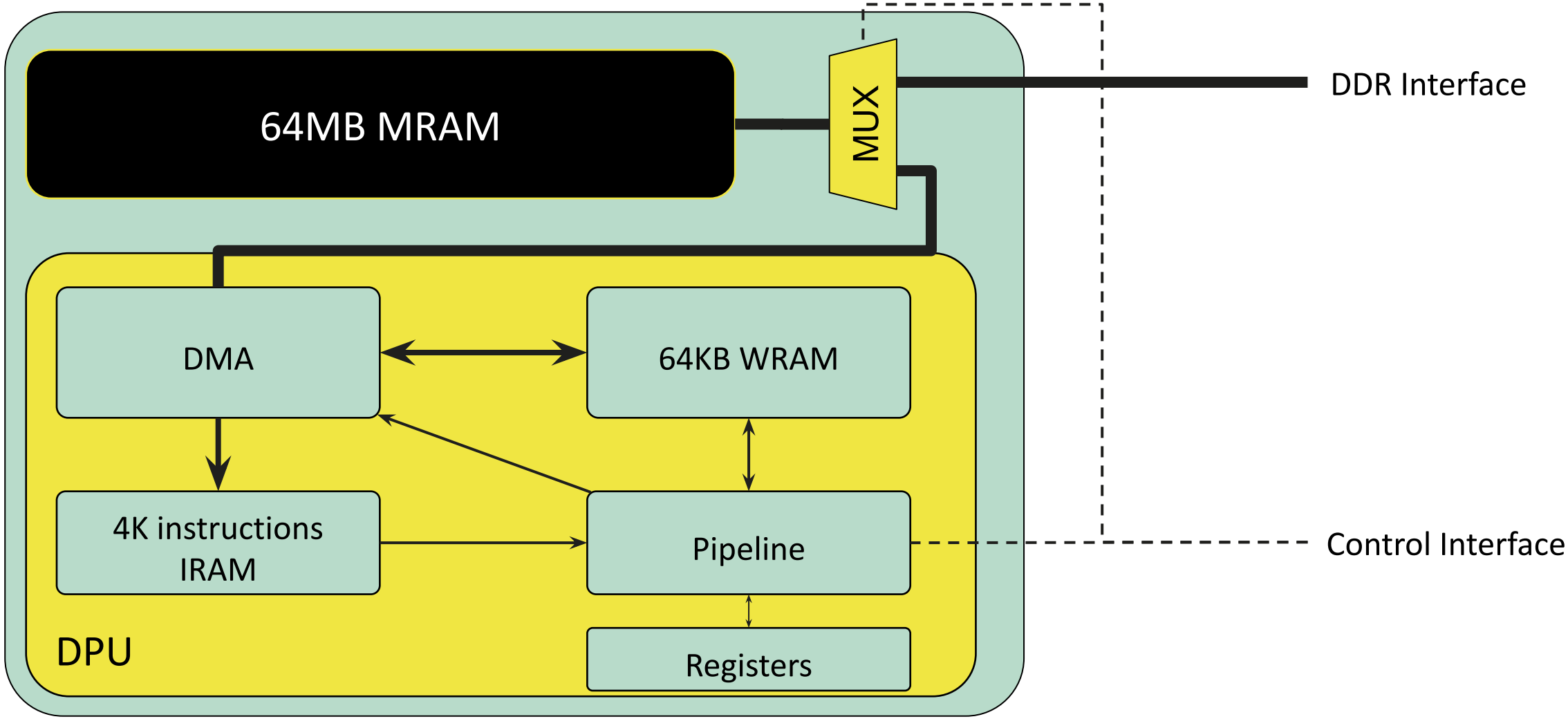
Integrating a processor into a DRAM



Integrating a processor into a DRAM

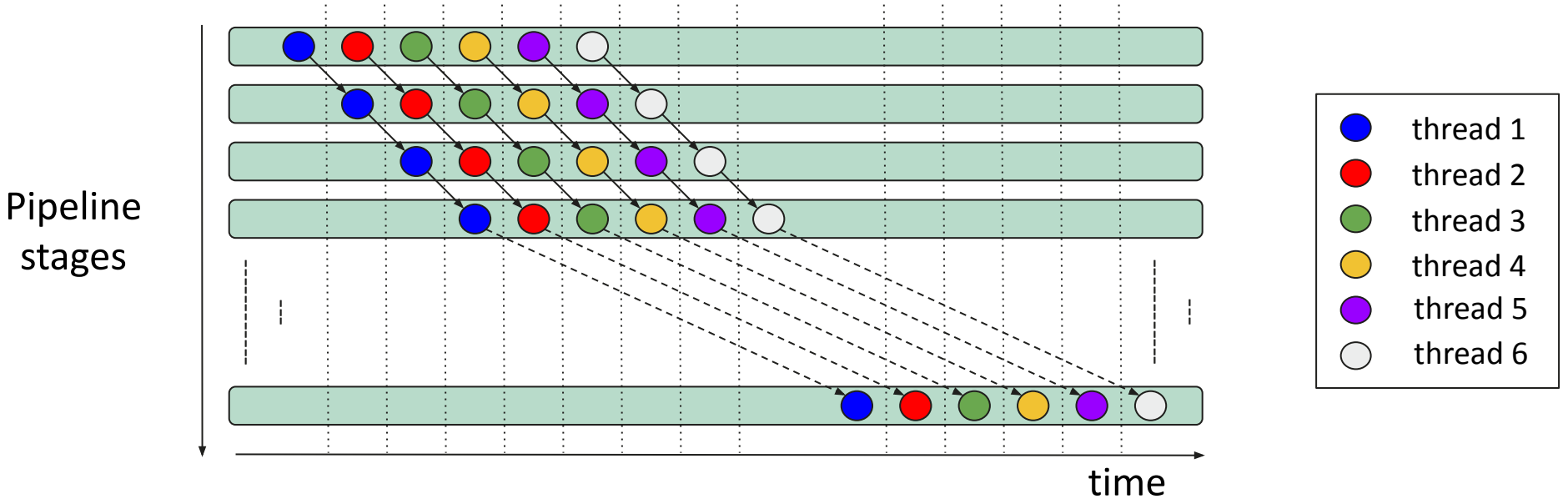


The DPU architecture

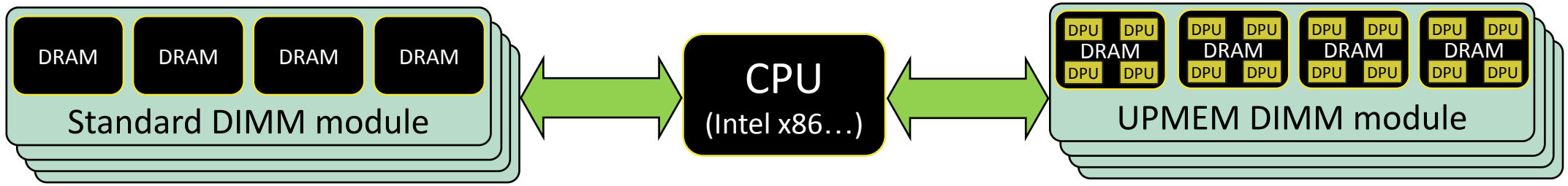


Pipeline properties

- 11-stage pipeline
- 24 hardware threads interleaved
- 32 registers per thread
  - 24 multi-purpose registers
  - 8 constant registers (zero, one, identifier of thread, etc)



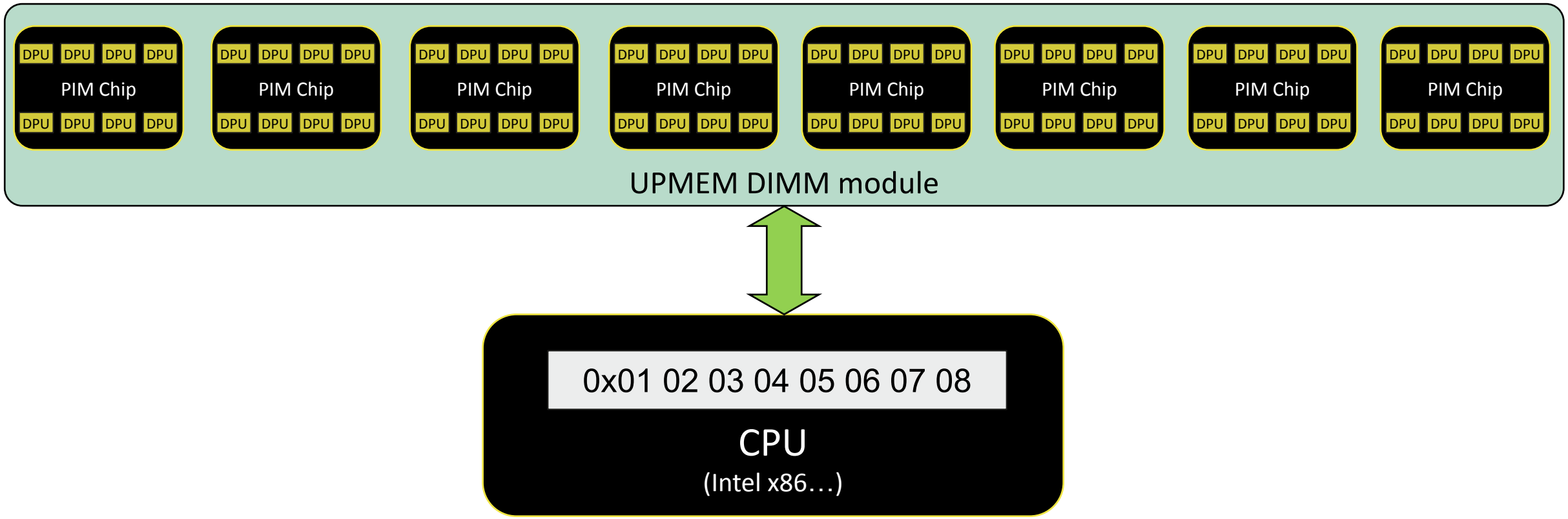
System overview



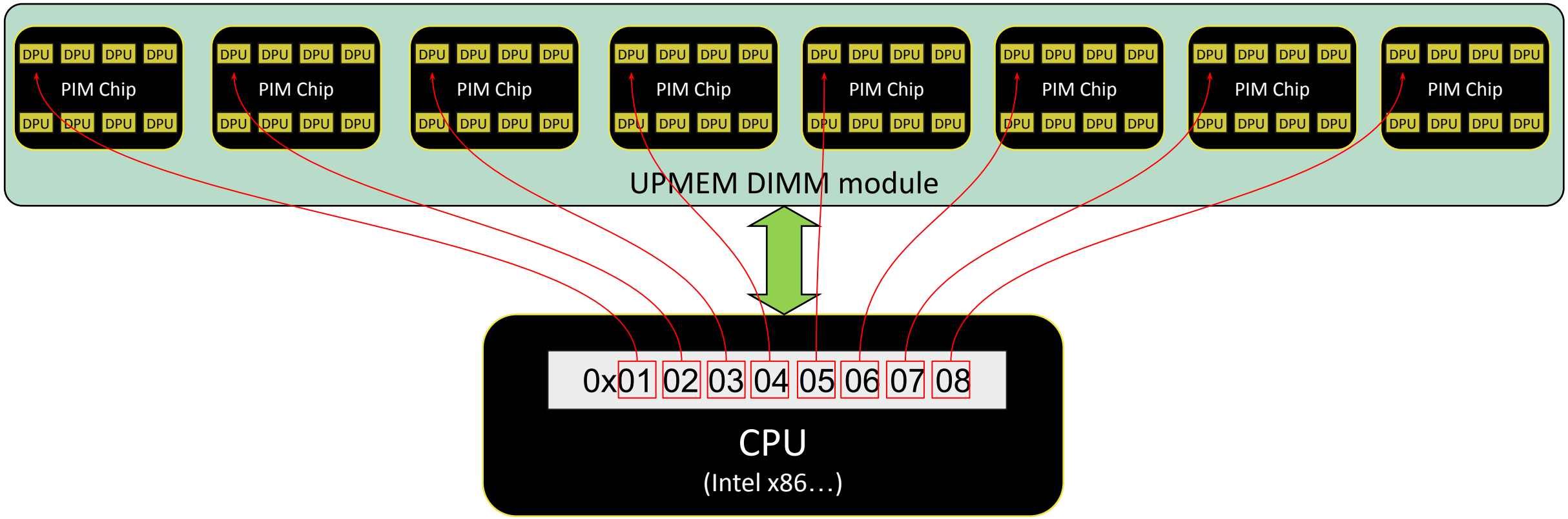
Typical workflow:

1. Init DPUs
2. Loop
   a. Copy data to DPU(s)
   b. Execute program on DPU(s)
   c. Copy data from DPU(s)

CPU/DPUs communication



CPU/DPUs communication



Programming the UPMEM PIM

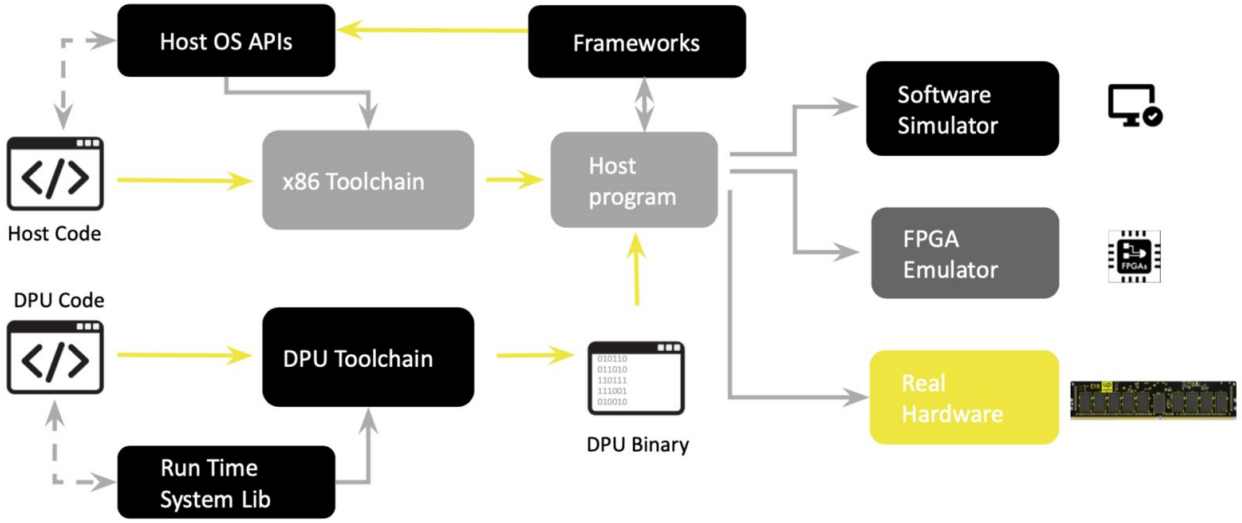
UPMEM SDK & Application Design Flow

- **CPU program**
  - compiled using the x86 toolchain
  - uses UPMEM's SDK host library
  - Allocate DPUs, load DPU program, boot DPUs, copy, asynchronous orchestration etc.

- **DPU program**
  - Written in C, subset of C library available
  - usually common to all DPUs (not mandatory)
  - compiled using dpu-clang (based on LLVM 12)
  - uses the DPU libraries for primitives like WRAM/MRAM management, thread synchronization (mutex etc.), perf counters etc.

- **Profiling & debugging tools**
  - DPU: dpu-lldb, dpu-trace, etc.
  - Host: dpu-profiling tool based on Linux perf

- **System**
  - Linux driver for x86 servers
  - Server BIOS binaries



The DPU ISA

- Proprietary RISC triadic instruction set
- **No FPU** : FP using a software library (IEEE-754)
  - ~200 cycles for a 32x32 float multiplication
  - consider quantization for FP applications
- No vectorized instructions (except cmpb4)
- 8x8 multiplication instruction, up to 32 ops for 32x32
- **Rich set of conditions for jump (examples next slide)**
- Assembly code analysis & optimization can be useful

- Operators (non-exhaustive):
  - Arithmetic: ADD, SUB, AND, OR, XOR
  - Loads/Stores: SD, SW, SH, SB, LD, LW, LH, LB
  - Shift/Rotate: LSL, LSR, ROL, ROR
  - Count bits: CLZ, CLO, CLS, CAO
  - Multiplication/Division: MUL_STEP, DIV_STEP, MUL
  - DMA loads/stores: SDMA, LDMA, LDMAI

The DPU ISA

16bit-integer multiplication

```
uint32_t mult(uint16_t a, uint16_t b) {
    return a * b;
}
```

```
mult:                                          // @mult
        move r2, r0
        mul_ul_ul r0, r1, r2, small, .LBB0_2
        mul_uh_ul r3, r1, r2
        lsl_add r0, r0, r3, 8
        mul_uh_ul r3, r2, r1
        lsl_add r0, r0, r3, 8
        mul_uh_uh r1, r1, r2
        lsl_add r0, r0, r1, 16
.LBB0_2:
        jump r23
```

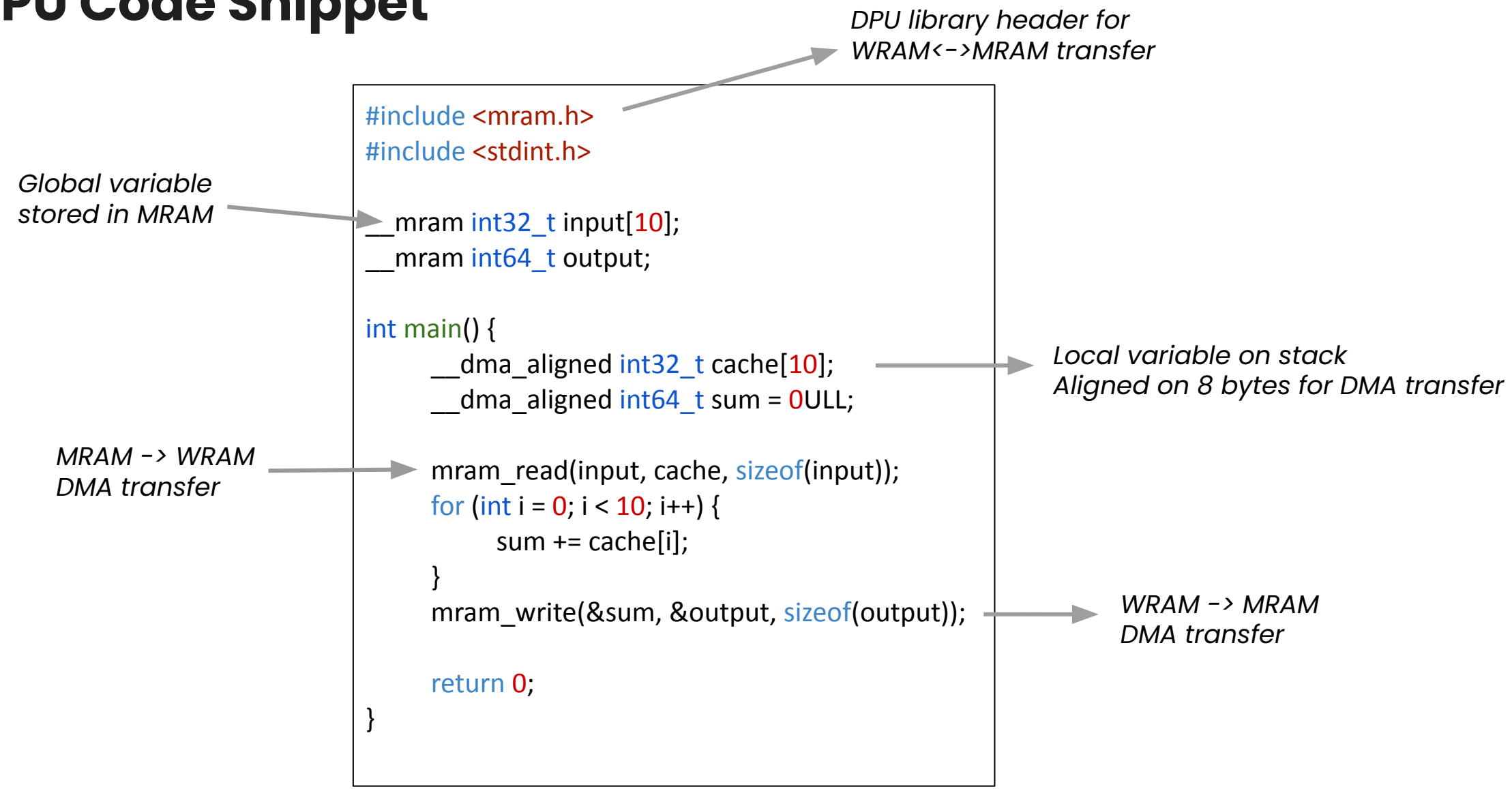
```
int sum(const int* data, int size)
{
    int val = 0;
    for(int i=size; i!= 0; i--)
        val += data[i];

    return val;
}
```

```
sum:
        move r2, r0
        move r0, 0
        jeq r1, 0, .LBB0_2
.LBB0_1:
        lsl_add r3, r2, r1, 2
        lw r3, r3, 0
        add r0, r3, r0
        add r1, r1, -1, nz, .LBB0_1
```

Reverse Loop

DPU Code Snippet




```
#include <mram.h>
#include <stdint.h>

__mram int32_t input[10];
__mram int64_t output;

int main() {
        __dma_aligned int32_t cache[10];
        __dma_aligned int64_t sum = 0ULL;

        mram_read(input, cache, sizeof(input));
        for (int i = 0; i < 10; i++) {
                sum += cache[i];
        }
        mram_write(&sum, &output, sizeof(output));

        return 0;
}
```

Host Code Snippet

Include C Host library header

```c
#include <dpu.h>

int array[10] = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9};

int main() {
    struct dpu_set_t dpu_set, dpu;
    DPU_ASSERT(dpu_alloc(1, "", dpu_set));
    DPU_ASSERT(dpu_load(dpu_set, "dpu_binary", NULL));

    DPU_FOREACH(dpu_set, dpu) {
        DPU_ASSERT(dpu_copy_to(dpu, "input", 0, array, sizeof(array)));
    }

    DPU_ASSERT(dpu_launch(set, DPU_SYNCHRONOUS));

    int64_t sum;
    DPU_FOREACH(dpu_set, dpu) {
        DPU_ASSERT(dpu_copy_from(dpu, "output", 0, &sum, sizeof(sum)));
    }
    return 0;
}
```

Allocate 1 DPU

Load DPU program

Copy the array content to the DPU variable named "input"

Boot the DPU program and wait for it to finish

Copy the DPU variable named "output" to the host variable sum

PIM Applications

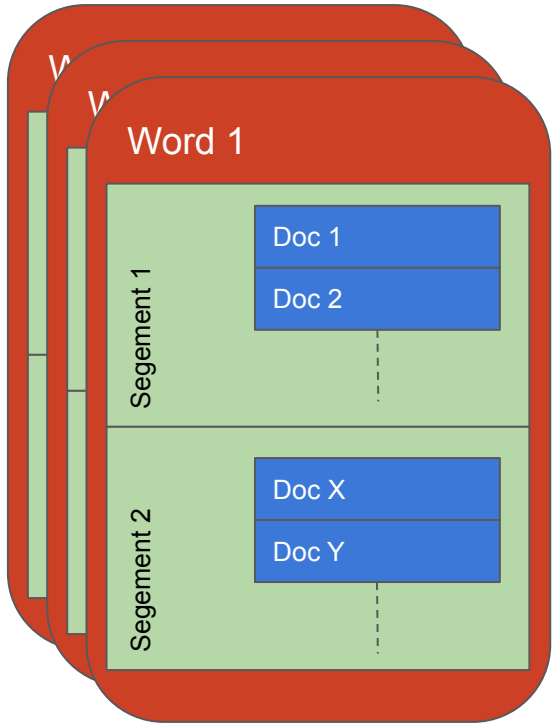
PIM Application Development

- **Which applications on UPMEM PIM ?**
  - Largely data-parallel (DPU + HW threads)
  - Low synchronizations, good load balancing
  - Memory-bound workloads (massive bandwidth of 2TB/sec on PIM)
  - Low operational intensity on CPU / cache inefficient
  - UPMEM PIM is throughput-oriented (batching)
  - PIM workload vs data transfer (large read-only database in PIM)
  - Consider algorithm redesign due to change of paradigm
- Maturing PIM applications is key to UPMEM's PIM product adoption
  - Customers do not want to develop the applications for us
- From algorithm benchmarking towards porting of standard libraries on PIM
- Application tightly coupled with SDK enhancements

Analytics : Index Search

- An index search engine identifies items in a database from keywords specified by the user (web pages, text documents, e-commerce product…)
- **UPIS**: Engine for exact phrase match https://github.com/upmem/usecase_UPIS
  - **› 600 queries/sec** with full PIM server on wikipedia
  - 30 queries/sec in Apache Lucene nightly benchmark
- **PIM Lucene**: extension of Apache Lucene https://github.com/upmem/pim-lucene

Inverted index



Analytics : Hash Join

- **Parallel hash-based join** on DPU
- 4G rows per table (32GB of random data)
- PIM-based join is **~9X faster** than CPU-based join (data fusion) using a server configuration of 2048 DPUs and 8 standard DIMMs

Left Table

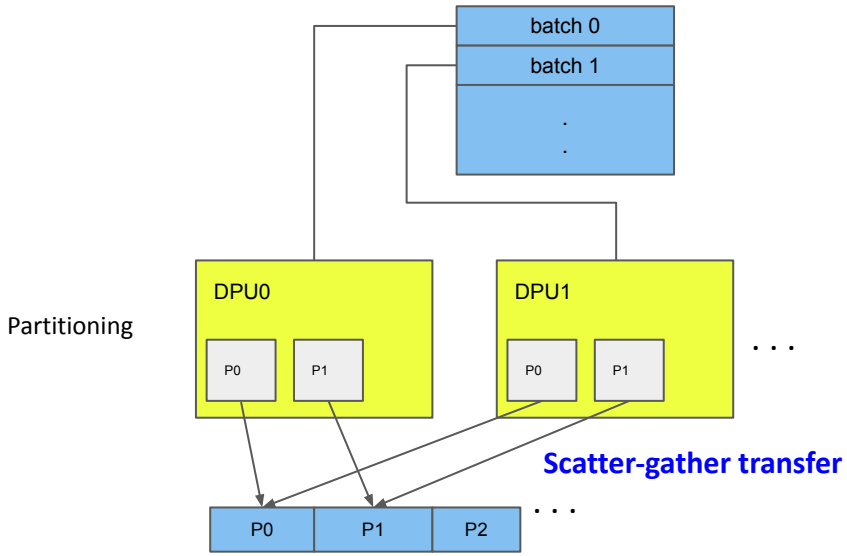
Date	CountryID	Units
1/1/2020	1	40
1/2/2020	1	25
1/3/2020	3	30
1/4/2020	2	35

Right Table

ID	Country
1	USA
2	Canada
3	Panama
4	Spain

Merged Table

Date	CountryID	Units	Country
1/1/2020	1	40	USA
1/2/2020	1	25	USA
1/4/2020	2	35	Canada
1/3/2020	3	30	Panama
null	*null*	*null*	Spain

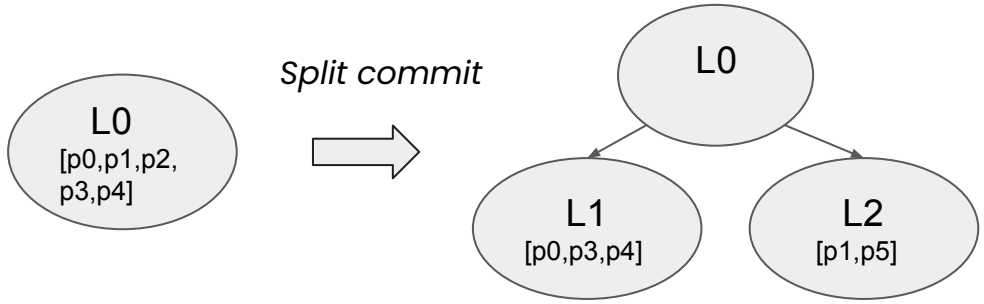


- Scatter-gather transfers (SDK 2023.1.0)
- Considering porting an OLAP DBMS to PIM (DuckDB)
- https://github.com/upmem/dpu_olap

up mem

ML : Decision Trees / K-means

- **CART** training implemented on DPU : builds a binary-search tree which represents a partitioning of the feature space
- **36X faster than CPU (scikit intel ext., criteo dataset)**
- Next step: XGBoost on PIM, throughput implementation



- **K-means** : partition the dataset into K distinct non-overlapping subgroups (clusters)
- **1.37× faster than CPU (scikit intel ext., criteo dataset)**
- **Paper:** Evaluating Machine Learning Workloads on Memory-Centric Computing Systems (ISPASS 2023)

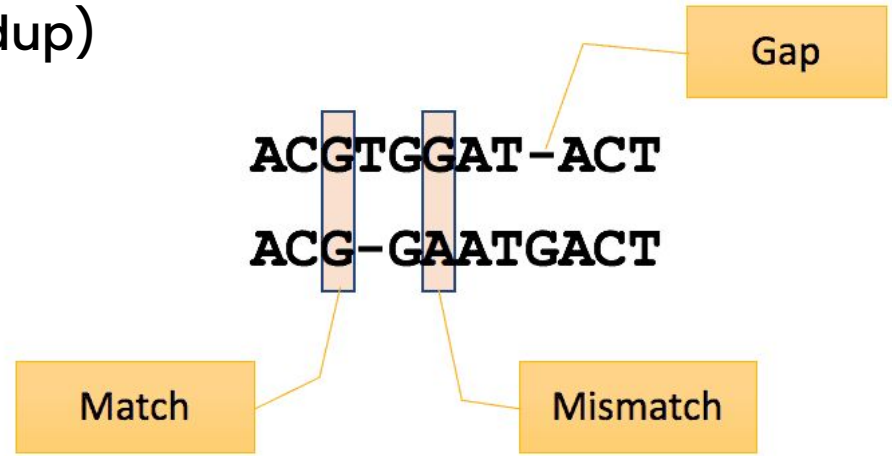
Genomics

- **Sequence alignment**

FASTQ file (read of 120 nucleotides):

```
@1
GGTTCACTGCAACCTCCGCCTCCCAGGTTCAAGCAATTATCCTGC
+
CCC0CGGGGGGGGGGGGGGGGGGGGGGGEGGGGGGGGGGD
@2
AGTGAATTCATTCAGAAAGATGAATTTCCTGTGTAATCTCATTATG
+
CC3ACEGGGGGGGGGGGCGGG1GCGGGGGGGGGGGGGG>G
```

- **UPVC**: Short reads alignment + variant calling
  https://github.com/upmem/usecase_UPVC
- **Long read alignment** : adaptive N&W algorithm (~9X speedup)
  https://github.com/upmem/usecase_dpu_alignment



- BWA (FM-index)
  - memory-bound algorithm but difficult to parallelize (too many synchronizations between DPUs)
- Pair-HMM (GATK variant calling)

up mem

Collaborative projects

EIC Accelerator

horizon europe

BioPIM

STRATUM 3D
3D Decision Support Tool for Brain Tumor Surgery

SustAInML

EUROPEAN UNION
EUROPEAN DEFENCE FUND

ΤΙΡΕΣΥΑΣ

bpifrance

ODYSSAI

PROJET FINANCÉ PAR L'ANR - PROJECT FUNDED BY THE ANR
ANR

GenoPIM
Processing-in-Memory for Genomics

upmem

BioPIM

Funded by
the European Union

Co-designing algorithms and data structures commonly used in bioinformatics together with several types of PIM architectures to obtain the highest benefit in cost, energy, and time savings.

3M€ project

SustainML

Funded by
the European Union

SustAInML

eProsima
The Middleware Experts

DFKI

Inria

IBM

Technische Universität Kaiserslautern

University of Copenhagen

up mem

Sustainable, interactive ML framework development for Green AI that will comprehensively prioritize and advocate energy efficiency across the entire life cycle of an application and avoid AI-waste.

4.3M€ project

up mem

STRATUM

Funded by
the European Union

STRATUM 3D
3D Decision Support Tool for Brain Tumor Surgery

ULPGC Universidad de Las Palmas de Gran Canaria · TU/e Eindhoven University of Technology · Region Stockholm · BSC Barcelona Supercomputing Center Centro Nacional de Supercomputación · up mem · Università di Pavia · Politécnica · SaludMadrid · RPTU Rheinland-Pfälzische Technische Universität Kaiserslautern Landau · FIISC · OPTOMIC producing quality · ecsa European Citizen Science Association

3D decision support tool for brain surgery guidance and diagnostics based on multimodal data processing through AI algorithms that will be integrated as an energy-efficient Point-of-Care computing tool.

10M€ project

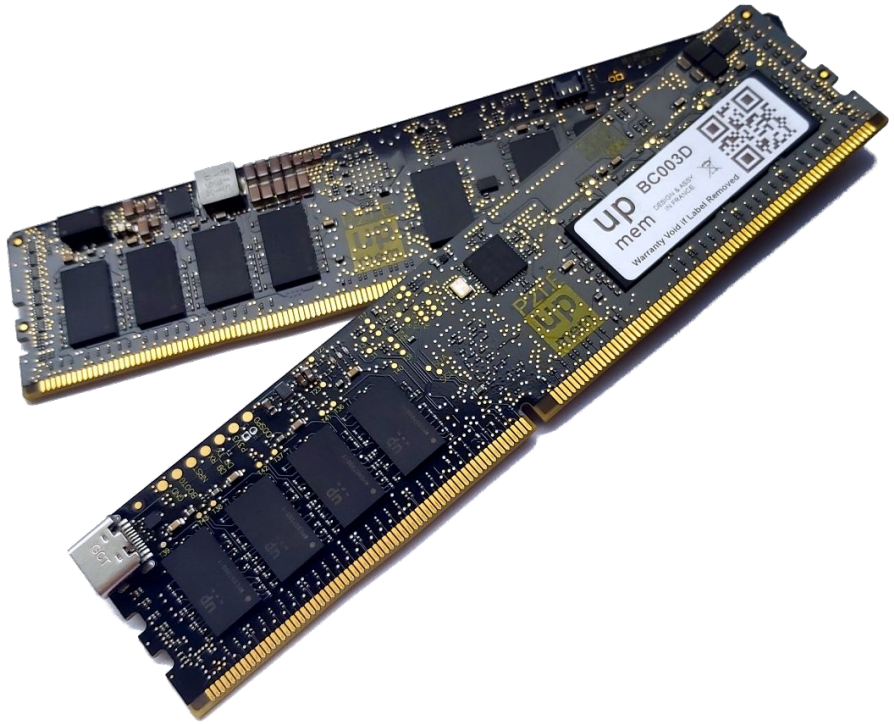
up mem

Hardware Roadmap

PIM DRAM Modules

Gen 1B (being released)

- Silicon bring up completed. System bring up in progress
- Frequency increased to 466 MHz or up to 40% lower power consumption at same frequency
- Host access to WRAM while the DPU owns the bank
- New HW monitoring features
- DPU switch off capability → Idle consumption ↘ by 90%



Gen 2

- Under development
- Enhanced control interface
- Host access to MRAM while the DPU owns the bank
- New operator integration
- Improved system integration

Gen AI

- Starting to work on a roadmap for a dedicated chip for AI applications such as LLM

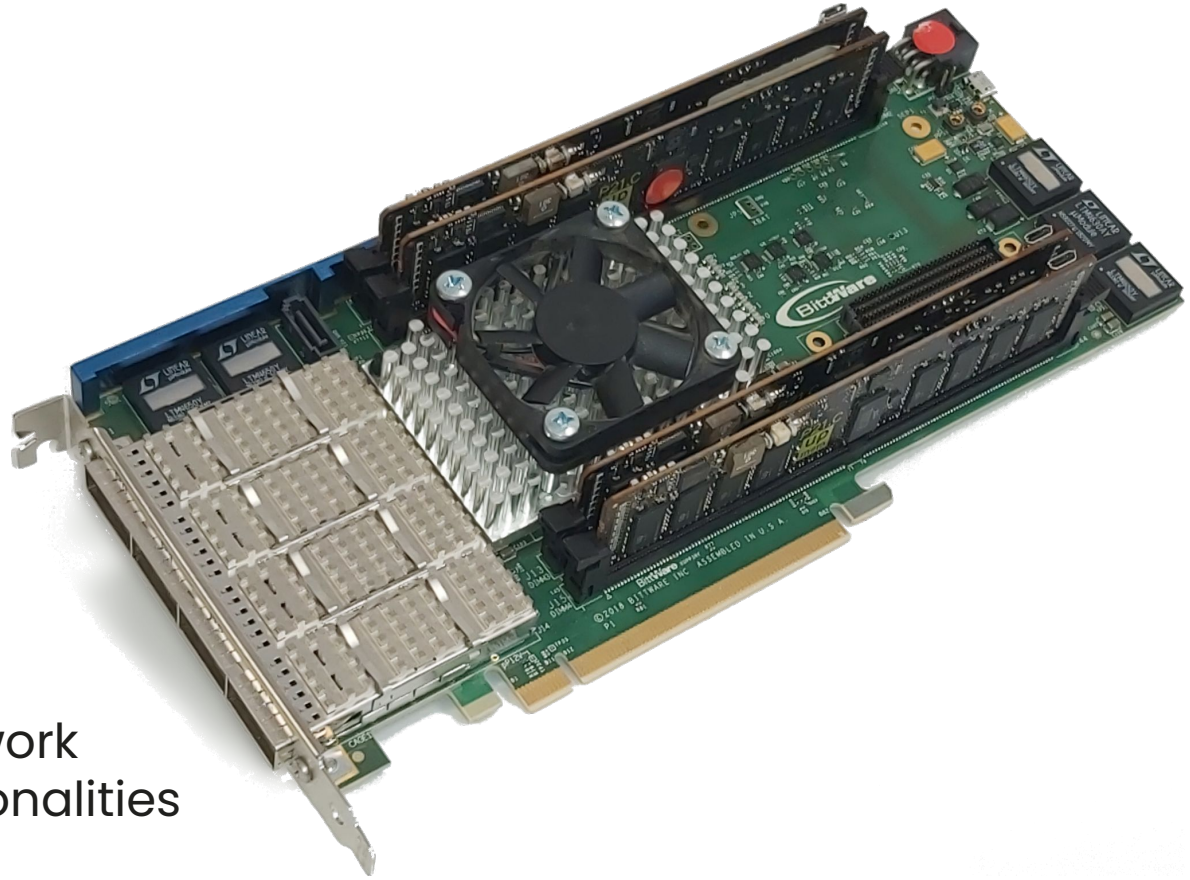
CXL or SoC attached integration

CXL Proof of Concept

- Based on a PCIe FPGA prototyping Card
- PIM friendly API embedded in the FPGA
- Simplifies system integration
- At the cost of higher BOM price and power consumption

PIM aware system

- Starting initiative with a major player in network devices to build a SOC with the same functionalities



BittWare XUP-P3R

up mem

Servers

From Skylake SP to Ice Lake SP

- Partnership with a major server manufacturer
- Moving from R&D servers to production servers
- 24 → 32 DIMM slots
- Increased number of memory channels
- Increased PIM and legacy memory density
- Released in coming months
- Early exploration of Sapphire Rapids SP

Open firmware implementation

- Partnership with another major server manufacturer
- Early exploration through a proof of concept

upmem

Cloud infrastructure

In Numbers

- 10 servers
- 56 teams
- 212 active users
- Almost 31 000 hours booked

Evolutions

- 1st spot where the aforementioned novelties will be released
- Service storage capacity (local disk, sftp for dataset pre-loading…)

Useful links

- Website
- Resource page
- Github
- SDK

Thank you

Yann FALEVOZ, In charge of lab relationship management
yfalevoz@upmem.com
Julien LEGRIEL, Technical leader on the SDK and applications on PIM.
jlegriel@upmem.com