Barcelona Supercomputing Center
Centro Nacional de Supercomputación

EXCELENCIA SEVERO OCHOA

BIMSA: Accelerating Long Sequence Alignment Using Processing-In-Memory

Author
Alejandro Alonso-Marín

Co-Authors:
Santiago Marco-Sola,
Ivan Fernandez,
Quim Aguado,
Juan Gomez-Luna,
Onur Mutlu,

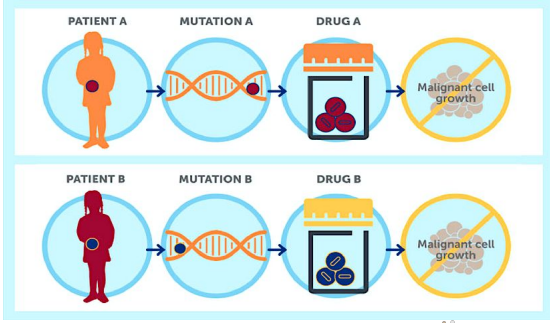
1. Introduction and Motivation
2. Background
3. PIM-enabled BiWFA Implementation
4. Experimental Results
5. Take Away Messages

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

1.1 Genomic Sequencing and Sequence Alignment

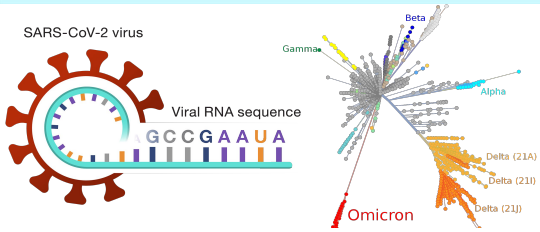
Sequencing an organism's genome enables:

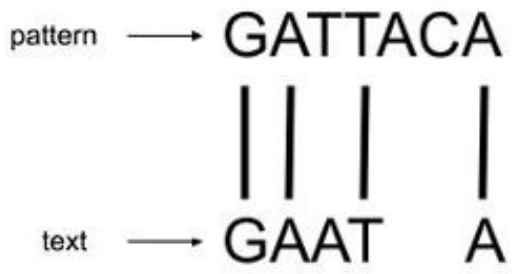
- **Personalized medicine/therapies**.
- **Tracing a real-time virus outbreak**.
- **Editing an organism genome**.



Genomic Sequencing requires Sequence Alignment which compares DNA/RNA sequences to obtain:

- **Path of operations** to transform one sequence into another (aka CIGAR).
- **Distance/error/score** (sum of the operation weights).





1.2 Sequence Alignment Algorithms

Classical sequence alignment algorithms are based on **Dynamic Programming** (DP) (e.g., Needleman-Wunsch, Smith-Waterman) which compute the full matrix (n×m).

- **Complexity O(n×m)** in memory and time.

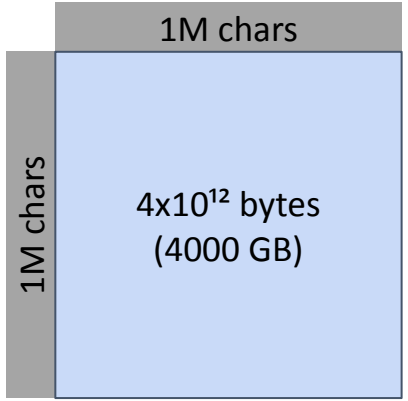
Wavefront Algorithm (WFA): computes the cells in increasing score **(s)**, avoiding suboptimal cells.

- **Complexity O(n×s)** in compute and **O(s^2)** in memory.

Bidirectional Wavefront Algorithm (BiWFA): computes **WFA** in both ends of the matrix.

- **Complexity O(n×s)** in compute and **O(s)** in memory.

All algorithms are a **memory bound** problem.



		G	A	A	T	A
	0	1	2	3	4	5
G	1	0	1	2	3	4
A	2	1	0	1	2	3
T	3	2	1	1	1	2
T	4	3	2	2	1	2
A	5	4	3	2	2	1
C	6	5	4	3	3	2
A	7	6	5	4	4	3

1. Introduction and Motivation
2. **Background**
3. PIM-enabled BiWFA Implementation
4. Experimental Results
5. Take Away Messages

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

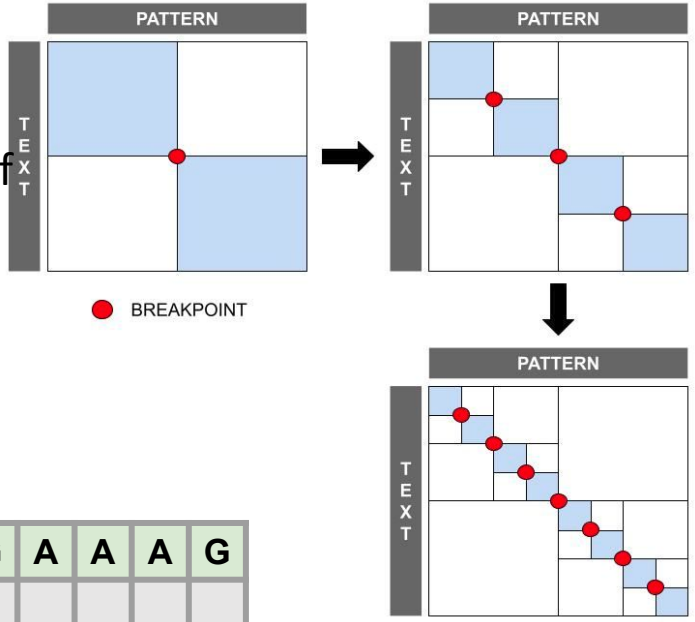
2.1 BiWFA Key Concept

1. Compute independent **WFA from both ends**.
2. Find where the two WFA collide/meet (breakpoint of the alignment).
3. **Recursively divide and conquer the problem.**

Breakpoint gives **distance**, offset and operation of the optimal path ⟶ All breakpoints = **Operation path**.

BiWFA needs **4 increasing arrays** (wavefronts) **O(s)**.

WFA stores intermediate wavefronts **O(s^2)**.



s=1	s=2
	4
2	5
	3

		T	G	G	A	A	A	G
	T	0	1	2				
	C	1	1					
	T	2		2				
	A				2			
	G							
	C							
	G							

FORWARD WAVEFRONT

		T	G	G	A	A	A	G
	T							
	C							
	T							
	A				2		2	
	G					2		2
	C						1	1
	G					2	1	0

REVERSE WAVEFRONT

s=2	s=1
5	
5	2
3	

2.2 Processing-In-Memory (PIM) Paradigm

Processing-In-Memory (PIM) is a paradigm that aims to alleviate memory bound problems.

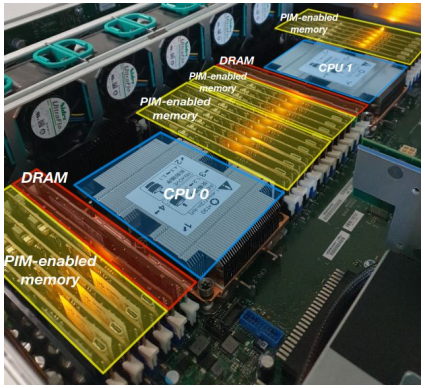
Key idea: place compute units close to data.

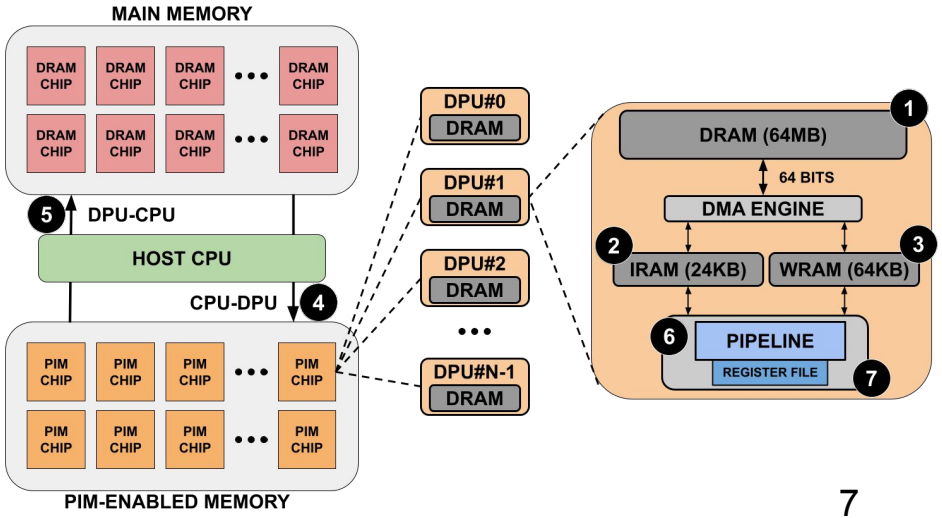
This enables:

- Exploiting **higher bandwidth** and **lower latency**.
- **Reducing energy** consumption.

We use the **UPMEM** PIM architecture.

1. Up to 2500 compute units (DPUs).
2. Up to 24 hardware threads.
3. 64MB DRAM memory (MRAM).
4. 64KB scratchpad memory (WRAM).
5. General purpose processor.





1. Introduction and Motivation
2. Background
3. **PIM-enabled BiWFA Implementation**
4. Experimental Results
5. Take Away Messages

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

3.1 BIMSA: BiWFA PIM-based Implementation

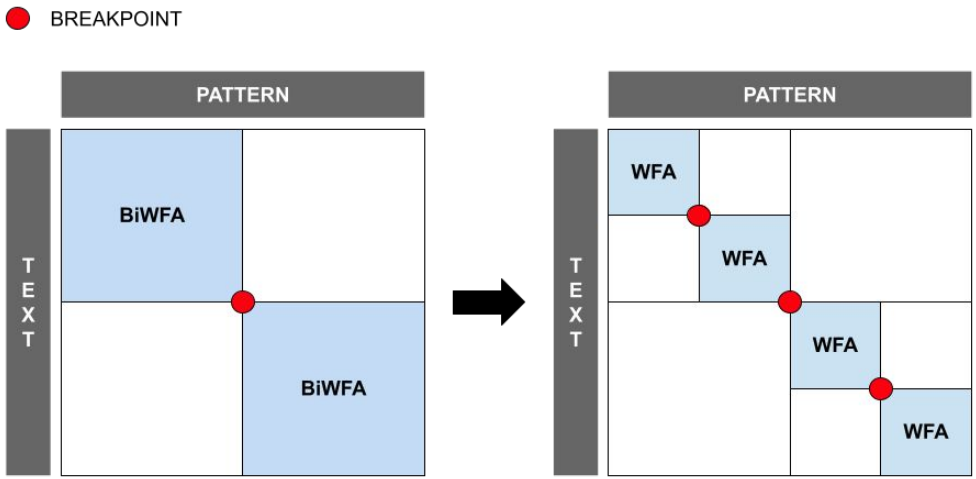
Challenge: Full-recursive BiWFA presents many corner cases.

Key idea: Execute BiWFA until the slices have a suitable size for WFA.

Leverage: BiWFA property → breakpoint gives distance.

For each thread (coarse-grain parallelism):

1. Compute BiWFA and find a **breakpoint**.
2. Slice sequences in half.
3. Compute BiWFA in both slices.
4. Iterate until distance equals a **threshold**.
5. Use WFA for the small slices **(base case)**.
6. Obtaining **partial CIGARs** from the base cases.
7. **Join partial CIGARs**.



3.2 BIMSA Optimizations: Base Case in Scratchpad

BiWFA steps use **4 scratchpad blocks independently** and **access DRAM** when the block is used.

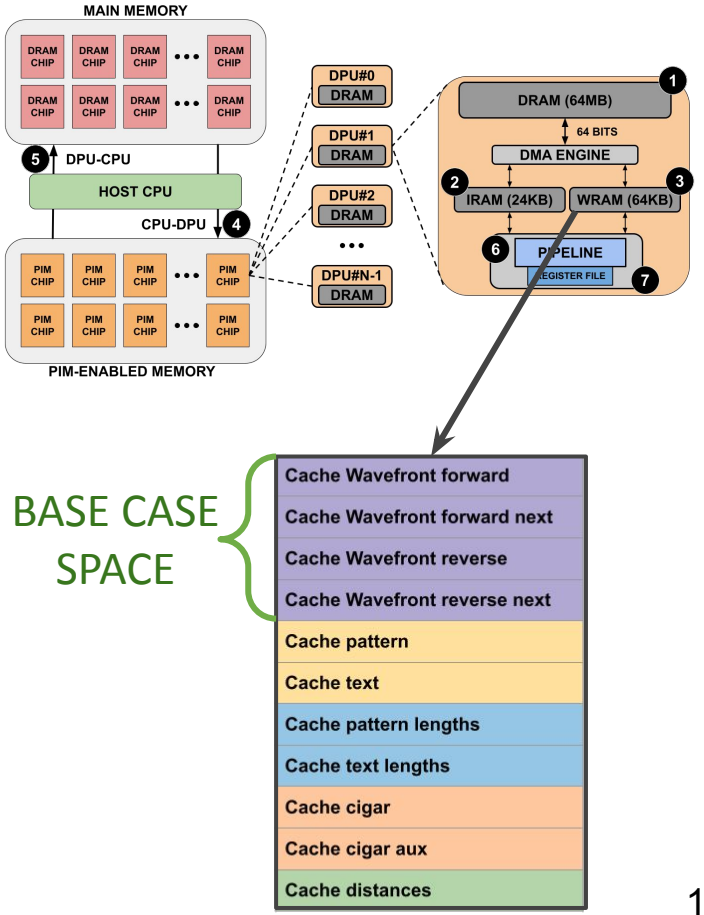
These 4 blocks are not needed for the base case.

- **Merge the 4 blocks** and use them for the base case.

WFA memory space is defined by distance.

- Use the **4 blocks size** as base case **threshold**.

We eliminate completely any DRAM access.

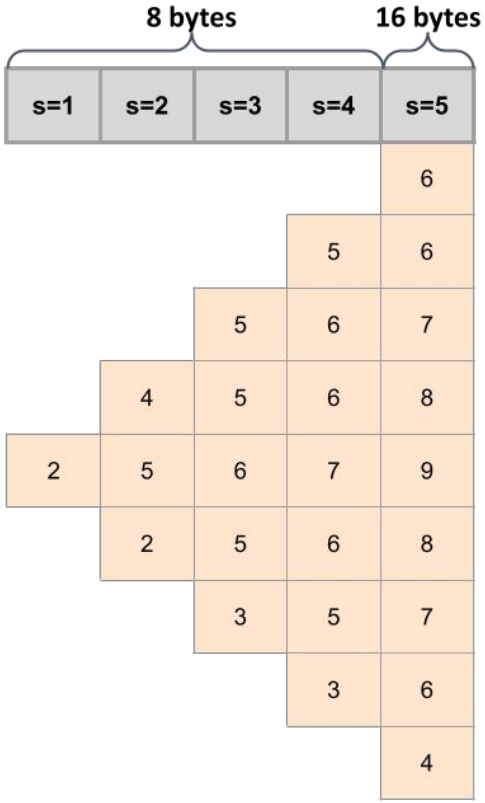


3.2 BIMSA Optimizations: Adaptive Transfers

BiWFA uses **increasing data structures** that reset its size after each breakpoint.

However, **we know** exactly the **amount of data to transfer** in each iteration.

- UPMEM allows to **indicate the transfer size** in each DRAM/scratchpad transfer.
- We define the scratchpad data structures to the maximum size we expect to transfer (2048 max).
- **Start from 8 byte** transfers and **increase x2** upon surpassing the current transfer size.
- **Reset to 8 bytes** after each breakpoint.
- With this mechanism we read minimum useless data.

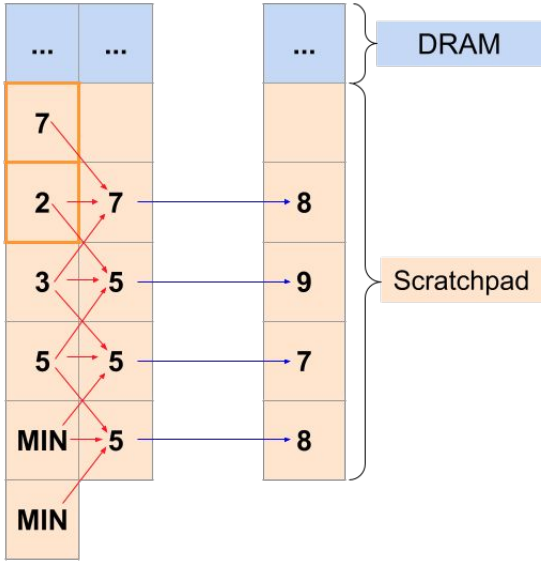


3.2 BIMSA Optimizations: BiWFA Steps Fusion

The original BiWFA makes use of vectorization ⟶ UPMEM does not support vectorization.

BiWFA uses **two main** computation **steps**. Applying them in large wavefronts require several memory transfers.

- UPMEM allows to explicitly **manage scratchpad** memory **blocks**.
- **Apply both steps on a scratchpad block.**
- Two elements are read twice per block.
- We reduce up to 40% the MRAM accesses.



3.2 BIMSA Optimizations: CPU Recovery

BIMSA is based on coarse-grain parallelism:

- No DPU intercommunication.
- High cost tasklet communication.

Datasets from real sequencers (e.g., Nanopore, PacBio) produce heterogeneous alignment times between tasklets and DPUs.

Solution (**BIMSA-Hybrid**):

- Alignment batching and different dpu_sets.
- Add distance/score limit to alignments O(ns) & finish larger alignments on CPU.

1. Introduction and Motivation
2. Main Tools Background
3. PIM-enabled BiWFA Implementation
4. **Experimental Results**
5. Take Away Messages

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

4.1 Experimental Setup

- Datasets
  - 9 simulated datasets (different sequence lengths and different error rates).
  - 4 real sequencer datasets (Illumina x2, Nanopore, PacBio).
- Applications
  - **CPU:** Official BiWFA implementation (WFA2lib).
  - **PIM:** state-of-the-art AIM implementations, WFA and NW (AIM-WFA, AIM-NW).
  - **BIMSA (Ours):** PIM - BiWFA implementation (UPMEM-v1B).
  - **BIMSA Next generation** projection (UPMEM-v1A) ⟶ Based on scalability results.
- Machines
  - **CPU**: Intel Xeon Silver 4215 2.50 GHz, 2 sockets of 8 cores each; 16 hardware threads.
  - **UPMEM-node**: 2556 DPUs 350 MHz, 24 threads.

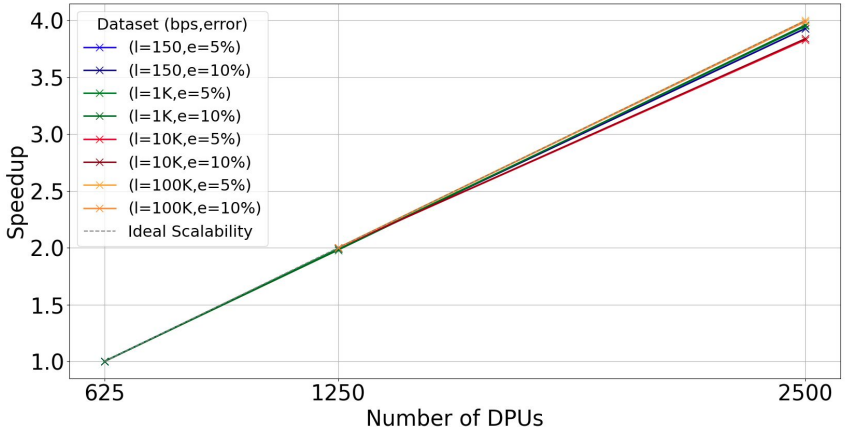
4.2 BIMSA Scalability (DPUs & Threads)

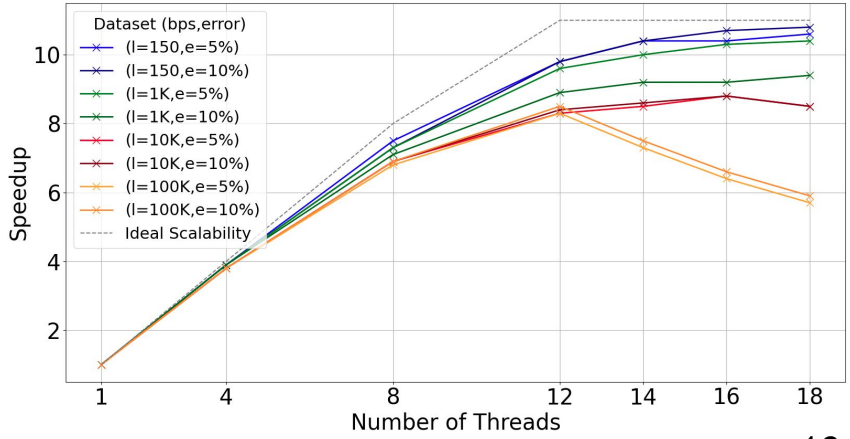
Optimal DPU scaling as long as the number of sequences is enough to feed all DPUs.

- **DPUs** are **not limited** by memory **bandwidth** like CPU (each DPU has its own memory bank).

11 threads are enough to obtain peak performance from the DPUs.

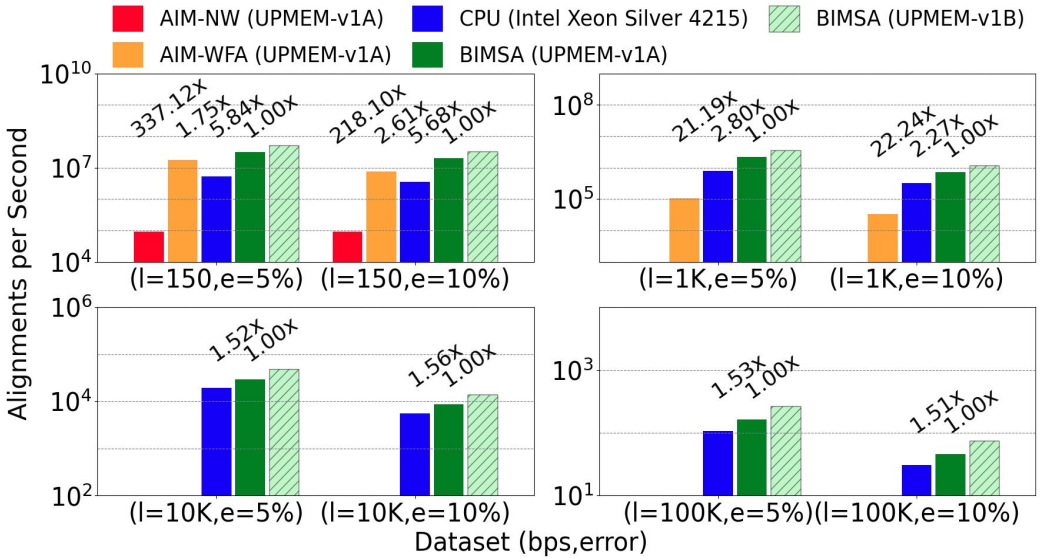
- Each thread can issue an instruction every **11 cycles** on the DPU pipeline.
- **Saturating** at **11 threads** means the **pipeline is** fully utilized.
- We start **saturating** at **8 threads** which indicates some **MRAM** access **saturation**.





4.3 PIM Performance Scaling in respect to CPU

- **AIM-NW** (DP-based on PIM) is outperformed by all the other implementations.
- **AIM-WFA** (WFA-based on PIM) is **unable to execute** long sequences and is outperformed by **BIMSA**.
- **BIMSA** (Ours) outperforms all other implementations and shows the **best scalability** for all datasets.
  Up to **22.23x** speedup against **AIM-WFA** and **5.84x** times against **WFA2lib.**
- **BIMSA projected** is almost able to double the performance in all cases.



4.4 PIM Performance for Heterogeneous Datasets

- **BIMSA** is **outperformed by CPU** in PacBIO and Nanopore **due to alignment heterogeneity**.
- **BIMSA-Hybrid outperforms CPU** by utilizing CPU recovery on the larger alignments.

Application	Illumina.150	Illumina.250	PacBio.CSS	Nanopore
AIM-NW	53.5	149.5	N/A	N/A
AIM-WFA	1.2	2.2	N/A	N/A
WFA2lib	0.8	1.2	9.2	206.1
BIMSA	0.6	1.1	182.4	314.5
BIMSA-Hybrid	0.6	1.1	7.6	105.6

New Feature: WRAM parallelism.

- Currently: Execute all batch alignments and stop to send to the CPU the larger alignments.
- Upcoming: Send large alignments to the CPU while computing the next alignment.

1. Introduction and Motivation
2. Background
3. PIM-enabled BiWFA Implementation
4. Experimental Results
5. **Take Away Messages**

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

5.1 Take Away Messages

- **Genomic sequencing** is a **fundamental** tool **in** modern **biology and healthcare**, which uses **sequence alignment,** a **memory bound** problem.

- **PIM** places computation closer to data, aiming the acceleration of **memory bound** problems such as **sequence alignment**.

- We present **BIMSA**, a PIM-aware optimized implementation of BiWFA for UPMEM. BIMSA achieves up to **22.23x** speedup against **AIM-WFA** and **5.84x** times against **WFA2lib**.

- **PIM** technology, while being immature, demonstrates promising results to accelerate memory-bound problems. We expect **significant improvements in the near future**.

Thanks for your attention!

Contact: **alejandro.alonso1@bsc.es**