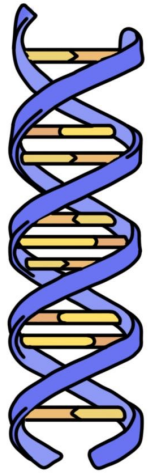
ABUMPIMP 2023: The 1st Minisymposium on Applications and Benefits of UPMEM commercial Massively Parallel Processing-In-Memory Platform
Full day minisymposium at EURO-PAR 2023

Protein Alignment on UPMEM PiM Architecture

Dominique Lavenier
GenScale, Univ. Rennes, IRISA/CNRS, INRIA - France

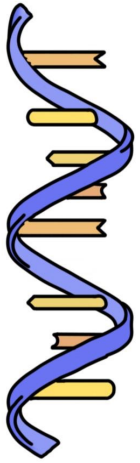
Central dogma of molecular biology

DNA

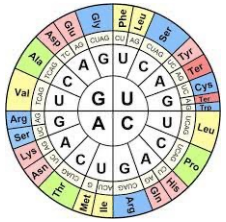


transcription

RNA

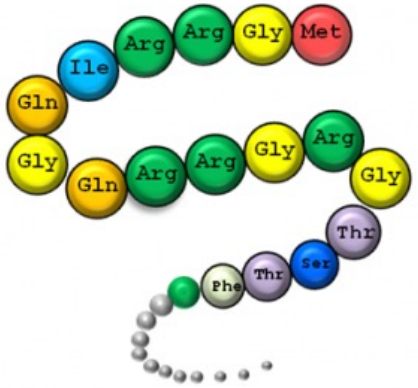


translation

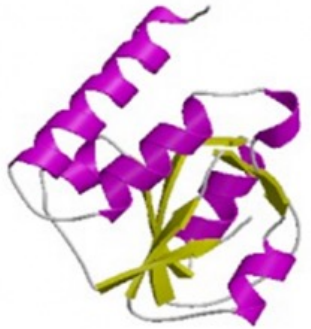


Genetic code

PROTEIN



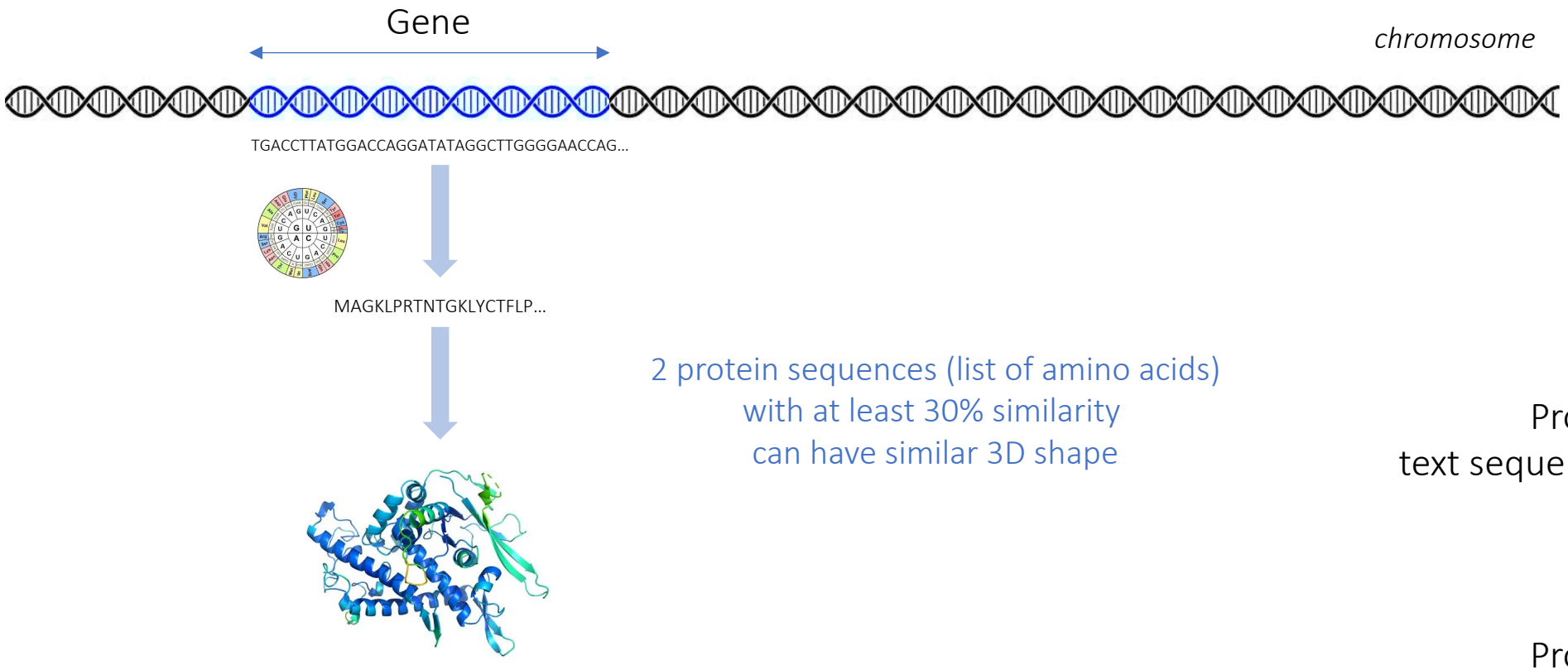
folding



Alphabet : nucleotides, 4 characters

Alphabet : amino acids, 20 characters

Text / Function relationship



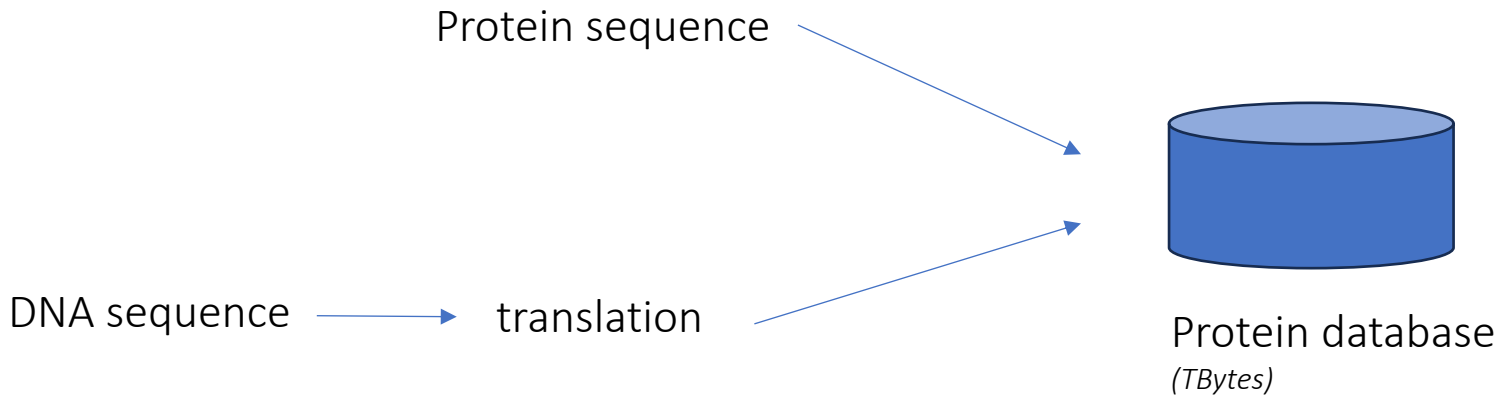
2 protein sequences (list of amino acids)
with at least 30% similarity
can have similar 3D shape

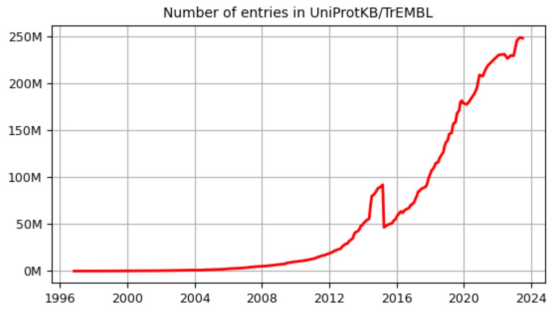
Protein
text sequence similarity

Protein
function similarity

The function of a protein is
determined by its 3D shape

Protein database search

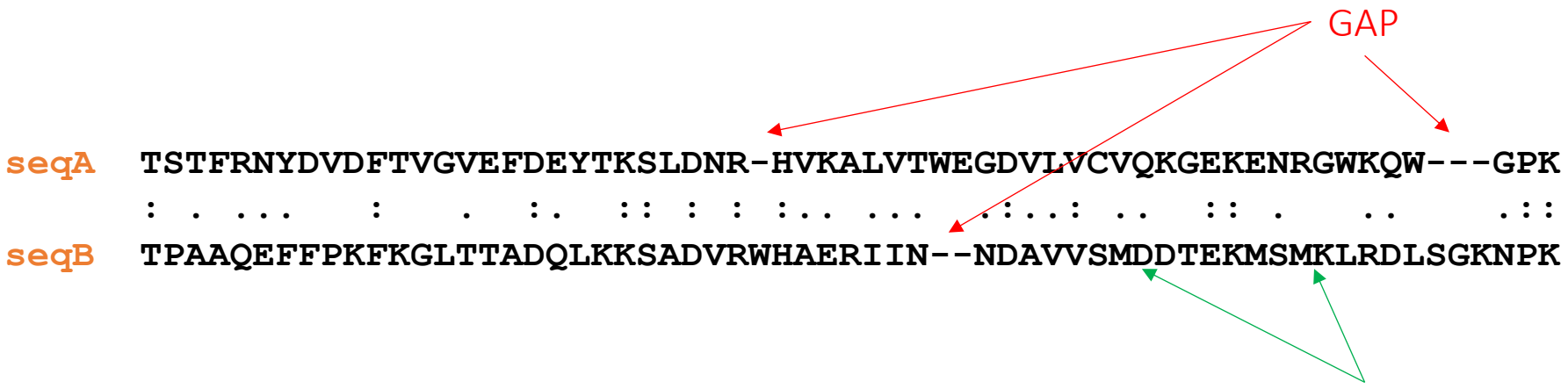




Execution time → hours/days/weeks of computation (metagenomic projects: billions of DNA sequences)

Querying results = list of alignments

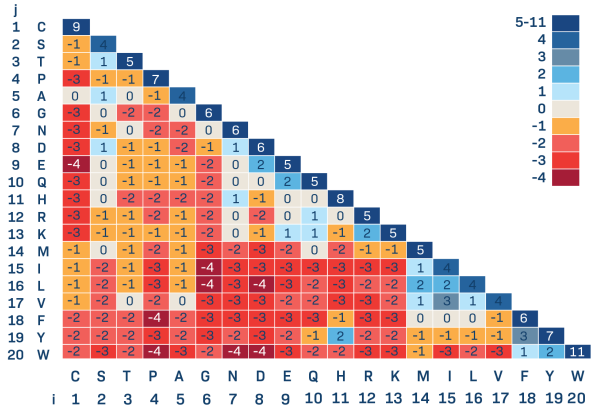
Alignments



Find the best match between 2 sequences using the minimum number of elementary edit operations

The score of an alignment is the cost of all edit operations

The substitution cost between 2 amino acids is given by a substitution matrix



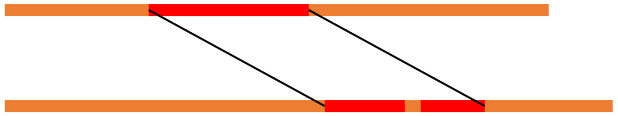
Alignments

2 types :

→ global alignment

To compare proteins of the same family

→ local alignment



To find similar regions

Computation of alignments

Seed & extend heuristics

Much faster than dynamic programing methods

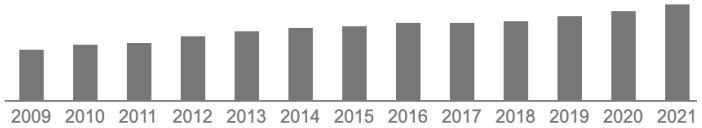
Reference software = BLAST

3 step process:

1. Search seed
2. Extend seed → hit
3. Compute alignment from hit

BLAST

Total citations Cited by 107157



Basic local alignment search tool
SF Altschul, W Gish, W Miller, EW Myers, DJ Lipman
Journal of molecular biology 215 (3), 403-410, 1990

seeds = words of 3 amino acids

→ Hypothesis:
An alignment can be built only if the 2 sequences share at least 2 compatible seeds within a given interval

Computation of alignments

Seed & extend heuristics

3 step process:

1. Search seed
2. Extend seed → hits
3. Compute alignment from hits

MEVF**PLT**GFAY
OLPQ**PLT**CQTP

No possible extension

MEVF**PLT**GFAYMCTGHPTYLIROWPNQPANPMVCTRE

KLMPRTYKSGHPVYLSROLPQ**PLT**CQTPLQANPER

Computation of alignments

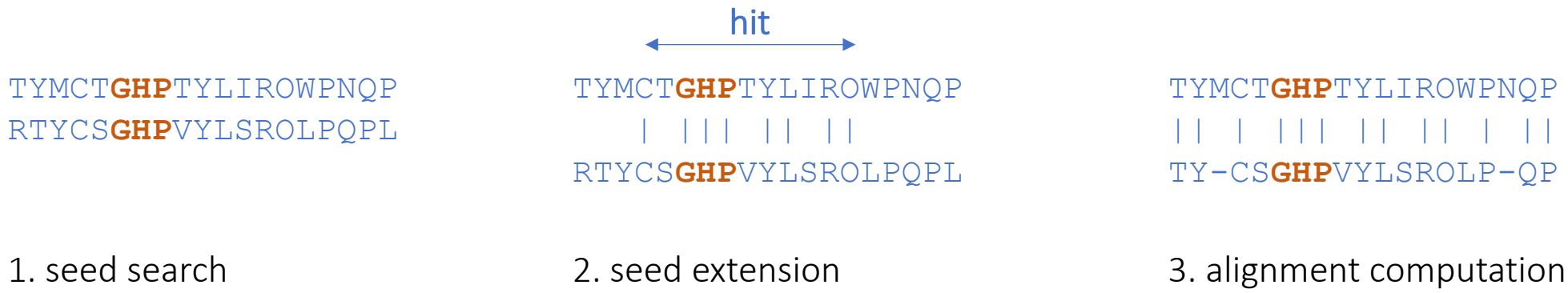
Seed & extend heuristics

3 step process:

1. Search seed
2. Extend seed → hits
3. Compute alignment from hits

```
MEVFPLTGFAYMCTGHPTYLIROWPNQPANPMVCTRE

KLMPRTYKSGHPVYLSROLPQPLTCQTPLQANPER
```

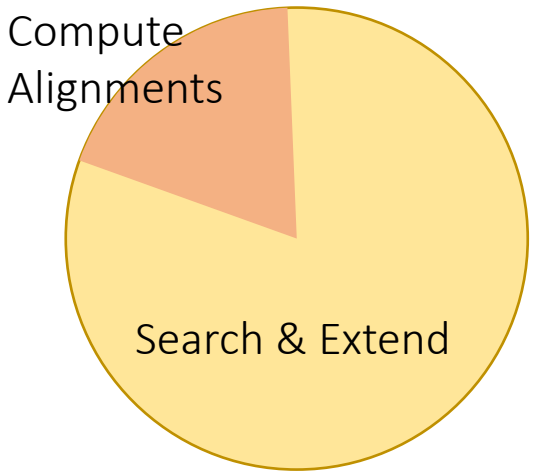


Computation of alignments

Seed & extend heuristics

3 step process:

1. Search seed
2. Extend seed → hits
3. Compute alignment from hits



% execution time

Most of the execution time is spent in step 1 & 2

PiM implementation

Pang: Protein Alignment with No Gap

First version of protein alignment on UPMEM PiM for feasibility study. On going work

Objectives

- Design a parallel implementation using both PiM and the host processor resources
  - PiM → DPU + MRAM
  - Processors → cores + legacy RAM

Seed-extend strategy:

1. Search seeds → DPU
2. Extend seed → hits
3. Compute alignments (no gap) → Host

Seed-extend strategy of Pang slightly different from the blast strategy

Same sensitivity

Database indexing

To speed up the “search” step (avoid systematic scan of the database)

Database = list of protein sequences

For each possible word of 3 characters over the protein alphabet, a list of tuples (n,x) is created:

n = protein sequence number

x = position of the word in the protein sequence

Example

Protein 1: TREQNES

Protein 2: MSTREP

TRE	(1,1), (2,3)
REQ	(1,2)
EQN	(1,3)
QNE	(1,4)
NES	(1,5)
MST	(2,1)
STR	(2,2)
REP	(2,4)

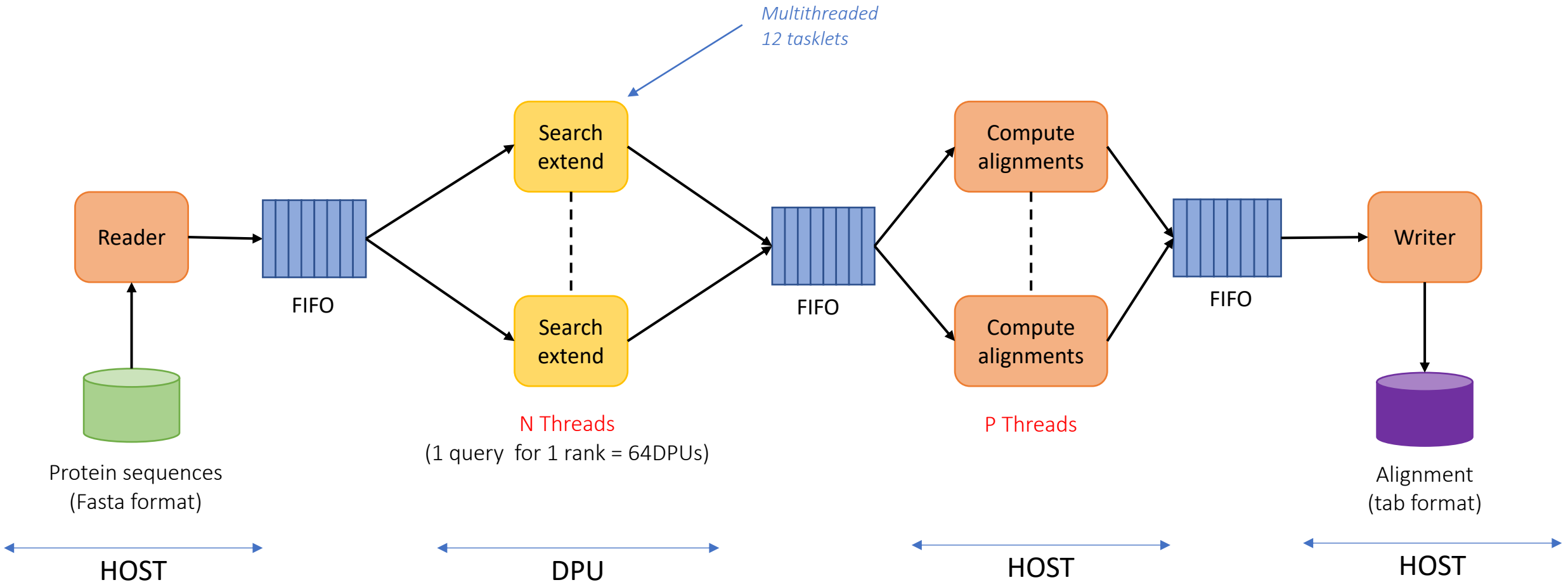
Index

- stored in the PiM memory
- fit in a single rank (64 DPUs, 4 Gbytes)
- split over 64 MRAM DPUs

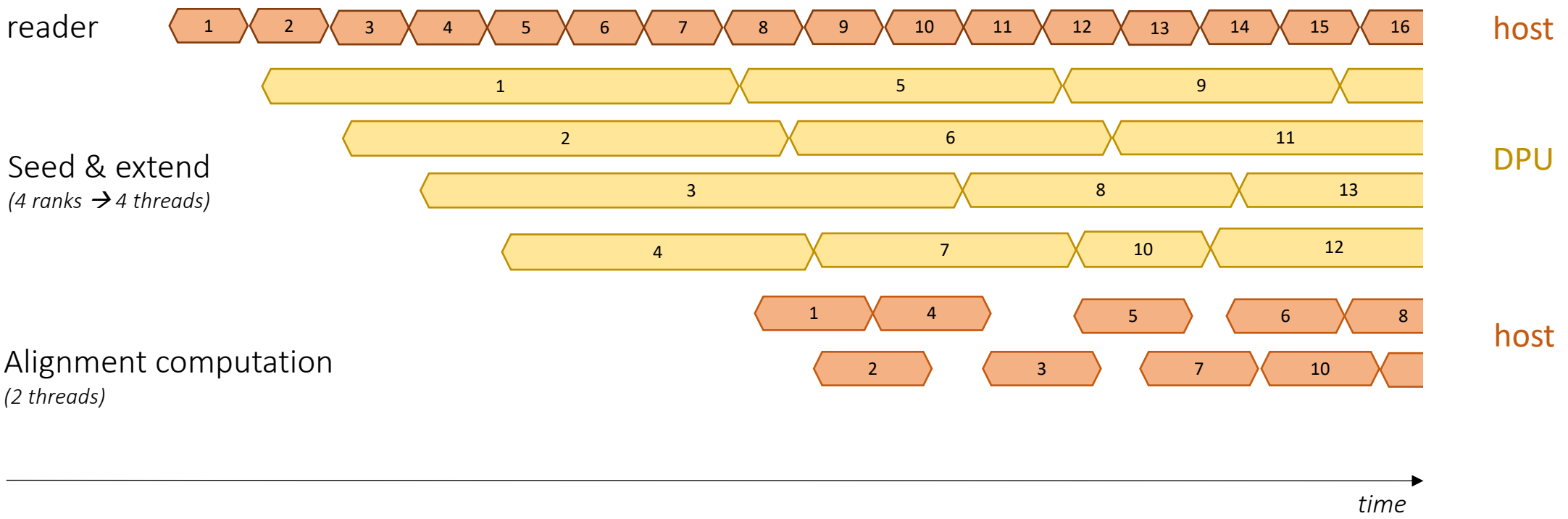
each MRAM contains part of the database

Database querying

DPU are dedicated to the search and extend steps
Host manage I/O and computes alignment



Synchronization



Experiments

Data sets

- Query
  - S1k.fa = 10^3 proteins
  - S10k.fa = 10^4 proteins
- Database = SwissProt, 563 000 proteins → index can fit on a rank (64 DPUs)

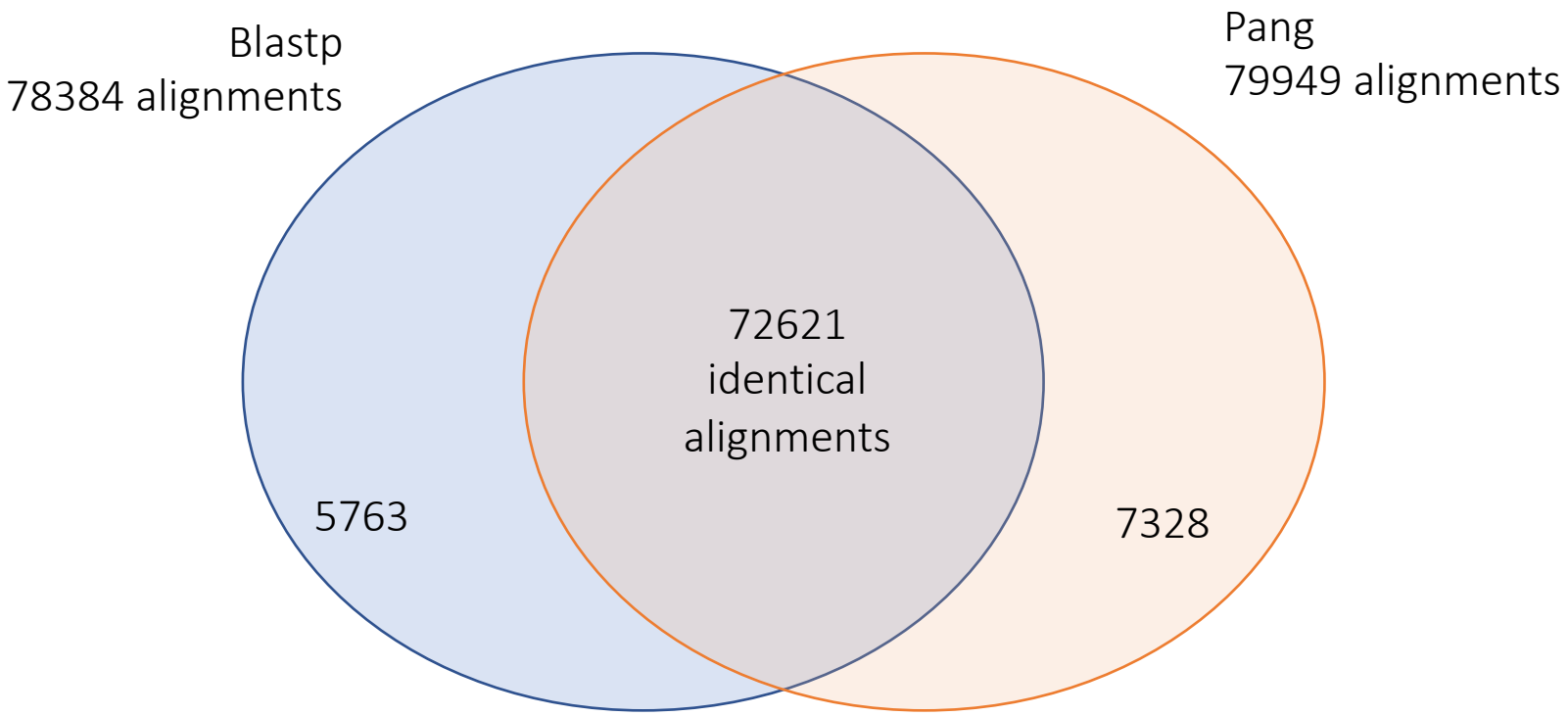
Hardware: UPMEM server

- CPU: 2 × Intel Xeon 4215 processeur, 2.5 GHz (16 cores)
- 256 Gbyes of standard memory
- 160 Gbytes of PiM memory (40 ranks of 64 DPUs)

Comparison with blastp (ungapped)

blastp -query s10k.fa -db sprot -ungapped -out align.blast -comp_based_stats F -evalue 1e-6 -seg yes -outfmt 6 -num_threads X

Pang vs Blastp sensitivity



Data : s1k.fa (10^3 proteins) vs SwissProt (563K proteins)

Pang speed-up vs Blastp

Rank

64 DPU on the same die
Optimized transfer

Run with 32 PiM ranks

Execution time: Blastp & Pang (in sec.) - 10^4 proteins

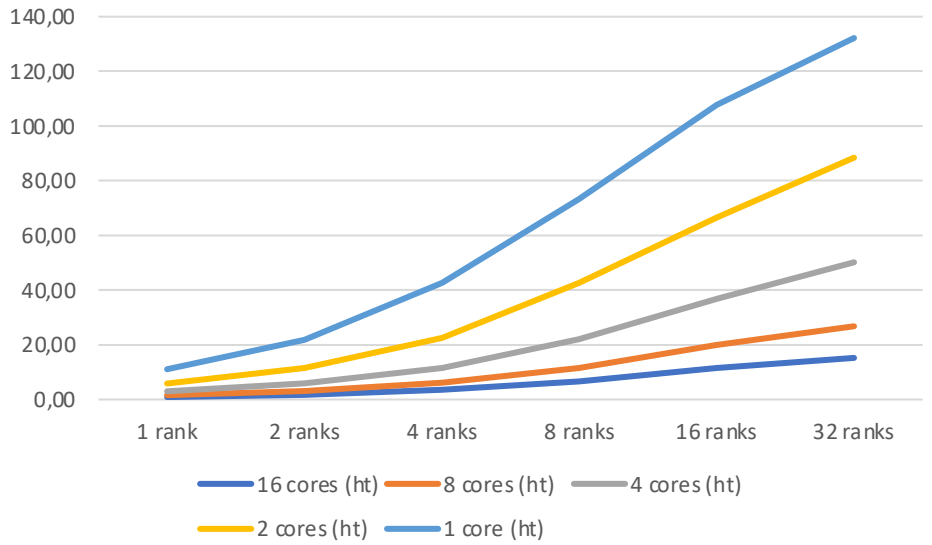
	blastp	Pang(1)	Pang (2)	Pang (4)	Pang (8)	Pang (16)	Pang (24)	Pang (32)
16 cores	844,00	938,14	473,26	240,98	127,15	73,37	62,51	55,46
8 cores	1490,00	939,83	475,31	243,12	128,75	74,42	61,13	55,62
4 cores	2821,00	939,72	475,00	242,66	128,34	76,87	62,06	56,22
2 cores	5495,00	939,05	475,32	242,94	128,63	82,50	67,48	62,13
1 core	10440,00	943,09	477,64	244,97	142,47	96,83	82,23	78,98

Pang time include database indexing
Taskset linux command is used to constrain the number of cores
Hyper threading is active

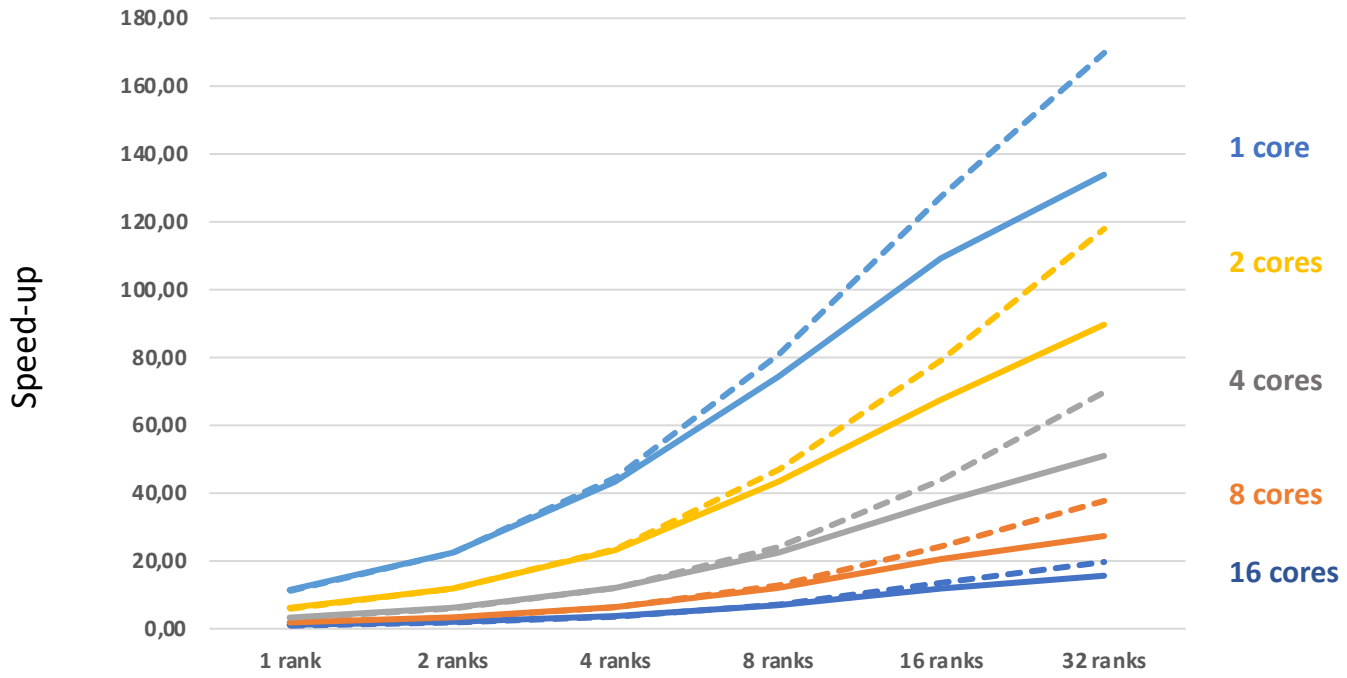
Speed-up = Blastp time / Pang time

Pang speed-up

	1 rank	2 ranks	4 ranks	8 ranks	16 ranks	32 ranks
16 cores	0,90	1,78	3,50	6,64	11,50	15,22
8 cores	1,59	3,13	6,13	11,57	20,02	26,79
4 cores	3,00	5,94	11,63	21,98	36,70	50,18
2 cores	5,85	11,56	22,62	42,72	66,61	88,44
1 core	11,07	21,86	42,62	73,28	107,82	132,19



Pang speed-up vs Blastp without database loading time



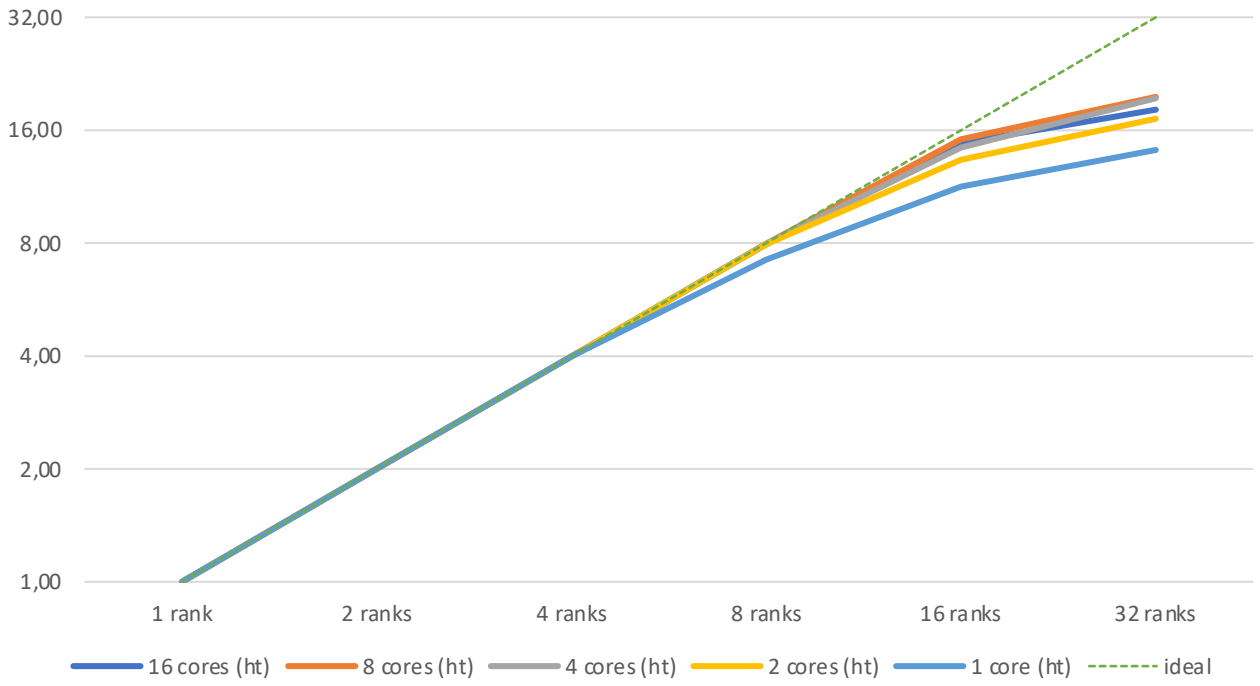
Without DB loading time

Speed-up = blast_time / (Pang_time – load_db_time)

With DB loading time

Speed-up = blast_time / (Pang_time)

Pang Scalability (without DB loading)



$$\text{Scalability} = \frac{\text{exec. Time 1 rank}}{\text{exec. Time N ranks}}$$

Conclusion

- Good speed-up on protein alignment can be achieved compared to commonly used software
  - DPU are well suited for string processing
    → No floating point operations, no multiplication, …

- Code need to be completely rewritten
  - Fine grained parallelism
  - Cannot implement specific functions due to DPU limitation (size of the scratchpad for example)

- Next
  - Extend analysis with other data sets
  - Energy saving analysis
  - Optimization with DIAMOND ideas
    - Software much faster than blastp, but less sensitive
  - Add gaps
    - Performances will slightly decrease
  - Nucleic vs Protein (blastx)

Thank you for your attention

Dominique Lavenier
lavenier@irisa.fr
GenScale, Univ. Rennes, IRISA/CNRS, INRIA - France

Université de Rennes

UMR IRISA

Inria