Westfälische
Wilhelms-Universität
Münster

High-Performance Probabilistic Record Linkage via Multi-Dimensional Homomorphisms

Ari Rasch, Richard Schulze, Waldemar Gorus,
Jan Hiller, Sebastian Bartholomäus, and Sergei Gorlatch

University of Münster, Germany
Epidemiological Cancer Registry North Rhine-Westphalia (NRW), Germany

Motivation

- *Probabilistic Record Linkage (PRL)* is the problem of identifying data records, e.g., in a database, that belong to the same real-world entity:

```
First Name: Marie
Last  Name: Smith
…
```

```
First Name: Mary
Last  Name: Smith
…
```

- PRL is used in many important areas such as the management of: hospitals, universities, and intelligence agencies.
- PRL has proven to very effective, but ·
- <u>In this work, we provide an implementation of PRL that:</u>
  - provides **high performance**,
  - is **portable** over different architectures (e.g., multi-core CPU, GPU, …).
- Our implementation is based on our approach of *Multi-Dimensional Homomorphisms*.
- Focus on a real-world case study — PRL as used in *Epidemiological Cancer Registry, Germany*.

Agenda

1. Probabilistic Record Linkage (PRL)
2. Multi-Dimensional Homomorphisms (MDH)
3. PRL as MDH
4. Using MDH Approach for Parallel PRL Implementation
5. Experimental Results.

Probabilistic Record Linkage

- PRL's basic idea is to use so-called *matching weights* `w(a,b)` of records a and b to identify duplicates.
- Matching weights are real numbers (typically between 1 and 100) that indicate the similarity between a and b:

```
‣ w(a,b) > UPPER_BOUND                        → duplicate
‣ w(a,b) < LOWER_BOUND                        → no duplicate
‣ LOWER_BOUND ≤ w(a,b) ≤ UPPER_BOUND → maybe duplicates (human review!)
```

Question: How is matching weight defined?

[1] Fellegi, Ivan P., and Alan B. Sunter. "A Theory For Record Linkage." *Journal of the American Statistical Association* 64, no. 328 (1969): pp. 1183-1210

Probabilistic Record Linkage

Matching weight `w(a,b)` is based on *matching/unmatching* probabilities of records a and b:

Matching Probability

$$m_i^x(a,b) = \mathbf{P}(\, a_i = b_i = x \mid (a,b) \in M \,)$$

Probability of:

- a and b refer to same real-world entity
- a and b coincide in attribute i (e.g., last name)
- attribute i is equal to x

Unmatching Probability

$$u_i^x(a,b) = \mathbf{P}(\, a_i = b_i = x \mid (a,b) \in U \,)$$

Probability of:

- a and b refer NOT to same real-world entity
- a and b coincide in attribute i (e.g., last name)
- attribute i is equal to x

Example:

- Last name "Dijkstra" has a low frequency.
- Last name "Smith" has a high frequency.

⇒ $m_{lastname}^{Dijkstra}(a,b) > m_{lastname}^{Smith}(a,b)$ and
$u_{lastname}^{Dijkstra}(a,b) < u_{lastname}^{Smith}(a,b)$

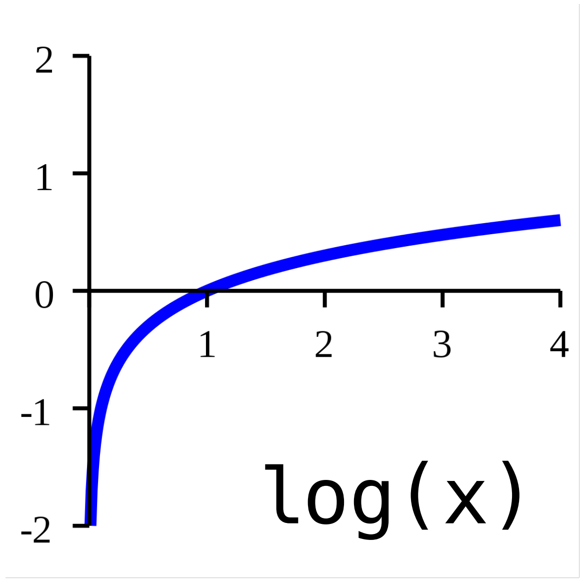
Probabilistic Record Linkage

The *matching weight in the i-th attribute* is computed as:

$$w_i(a,b) = \begin{cases} log(\frac{m_i^x(a,b)}{u_i^x(a,b)}) & : a_i = b_i \wedge x = a_i \\ log(\frac{1-m_i^x(a,b)}{1-u_i^x(a,b)}) & : a_i \neq b_i \wedge x = a_i \end{cases}$$

Illustrative: `w_i(a,b)` is defined to be high when

- attributes coincide that have high matching probability and low unmatching probability.
- attributes not coincide that have low matching probability and high unmatching probability.



The *matching weight w(a,b) is*:

$$w(a,b) = \sum_{i=1}^{N} w_i(a,b)$$

In words:

"Mary Dijkstra" and "Marie Dijkstra" are rather duplicates than "Mary Smith" and "Marie Smith" → last name "Smith" has a higher frequency than last name "Dijkstra".

Probabilistic Record Linkage

Summary: are records a and b duplicates?

1. Compute **matching probabilities** m_i^x and **unmatching probabilities** p_i^x

Matching Probability

$m_i^x(a,b) = \mathbf{P}(\, a_i = b_i = x \mid (a,b) \in M \,)$

Probability of:

- a and b refer to same real-world entity
- a and b coincide in attribute i (e.g., forename)
- attribute i is equal to x

Unmatching Probability

$u_i^x(a,b) = \mathbf{P}(\, a_i = b_i = x \mid (a,b) \in U \,)$

Probability of:

- a and b refer NOT to same real-world entity
- a and b coincide in attribute i (e.g., forename)
- attribute i is equal to x

2. Compute **matching weights in i-th attribute** w_i(a,b)

$$w_i(a,b) = \begin{cases} log(\frac{m_i^x(a,b)}{u_i^x(a,b)}) & : a_i = b_i \wedge x = a_i \\ log(\frac{1-m_i^x(a,b)}{1-u_i^x(a,b)}) & : a_i \neq b_i \wedge x = a_i \end{cases}$$

3. Compute **matching weight** w(a,b)

$$w(a,b) = \sum_{i=1}^{N} w_i(a,b)$$

Epidemiological Cancer Registry

- In the *Epidemiological Cancer Registry (ECR)*, PRL is used for avoiding duplicate entries in their patient data base.
- Duplicates can occur when same patient is accidentally registered by different registration offices under different names (e.g., Mary Smith vs. Marie Smith).

PRL in ECR:

- Patients are represented using 14 attributes.
- ECR uses *averaged matching probability* m_i (instead of matching probability m_i^x):

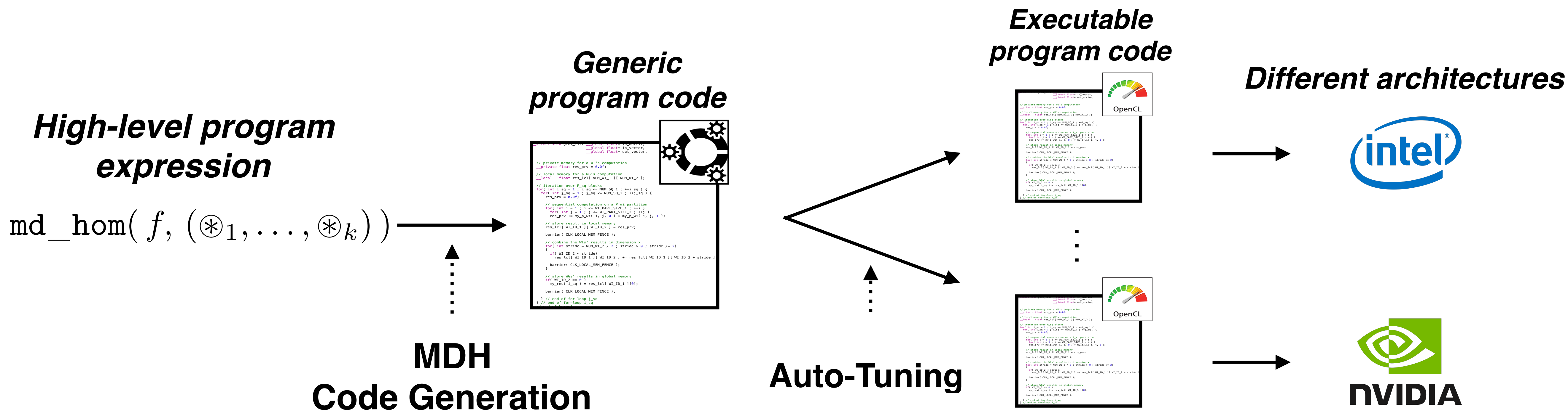
$$m_i = \mathtt{avg}_x \; m_i^x$$

e.g., $m_{forename}$ is probability that two records referring to same real-world entity have same (arbitrary) forename → this is because data bases with duplicates are rare but required for computing m_i^x.

No.	Attribute	m_i
1	Surname 1	0.975
2	Surname 2	0.975
3	Surname 3	0.975
4	Forename 1	0.975
5	Forename 2	0.975
6	Forename 3	0.975
7	Birth name 1	0.975
8	Birth name 2	0.975
9	Birth name 3	0.975
10	Day of birth	0.99
11	Month of birth	0.99
12	Year of birth	0.99
13	Gender	0.999
14	Municipality key	0.9

Multi-Dimensional Homomorphisms

Our approach of Multi-Dimensional Homomorphisms allows to conveniently generate high-performance code targeting multi- and many-core architectures:



Representation of important applications in our approach:

Linear Algebra

```
DOT  = md_hom( *, (+)          ) o viewBLAS
GEMV = md_hom( *, (++, +)      ) o viewBLAS
GEMM = md_hom( *, (++, ++, +)  ) o viewBLAS
```

Tensor Contractions

```
TC = md_hom( *, (++,…,++ , +,…,+) ) o viewTC<cD,sD>
```

Convolutions

```
Gaussian = md_hom( conv, (++, ++)             ) o viewStencil<C>
MCC      = md_hom( conv, (++, ++, ++, ++, +)  ) o viewStencil<C>
```

Speedups up to >4x as compared to state-of-the-art approaches!

Multi-Dimensional Homomorphisms

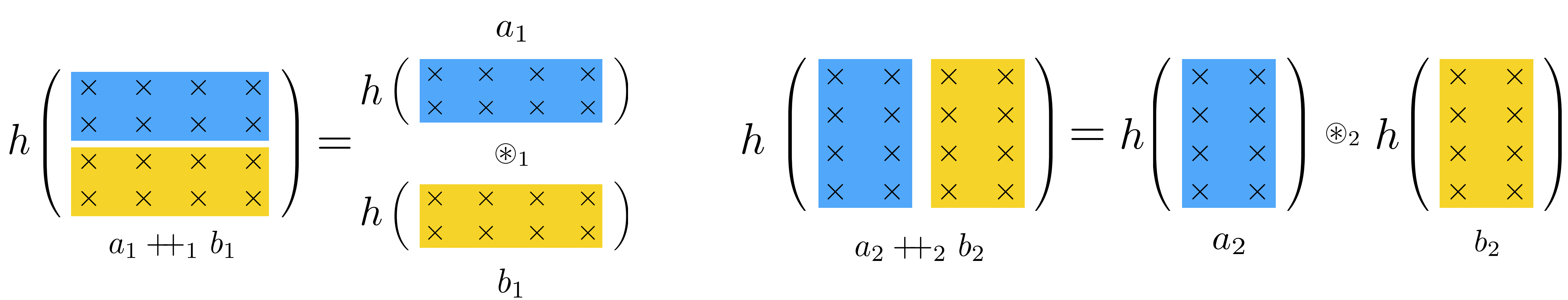
Our goal for PRL: express it as MDH to generate high-performance code for CPU and GPU.

Definition: [*Multi-Dimensional Homomorphisms* [2]]

Let T and T' be two arbitrary types. A function $h : T[\,N_1\,]\ldots[\,N_d\,] \to T'$ on d-dimensional arrays is called a *Multi-Dimensional Homomorphism (MDH)* iff there exist *combine operators* $\circledast_1, \ldots, \circledast_d : T' \times T' \to T'$, such that for each $k \in [1, d]$ and arbitrary, concatenated input MDA $a \mathbin{+\!\!+}_k b$:

$$h(\ a \mathbin{+\!\!+}_k b\ ) \ = \ h(a) \circledast_k h(b)$$

Examples (2D):



[2] Rasch, Ari, and Sergei Gorlatch. "Multi-Dimensional Homomorphisms and Their Implementation in OpenCL." *International Journal of Parallel Programming* 46, no. 1 (2018): 101-119.

Multi-Dimensional Homomorphisms

MDHs have a uniform representation:

Proposition:

Every MDH h is completely determined by its combine operators $\circledast_1 \dots \circledast_d$ and its action f on singleton arrays (i.e., $h(a) = f(a[0] \dots a[0])$).

Illustrative (2D):

$$h(a) \;=\; \overset{\xrightarrow{\quad\quad\quad\quad \circledast_2}}{\begin{pmatrix} f(a[0][0]) & \dots & f(a[0][n]) \\ \vdots & & \vdots \\ f(a[m][0]) & \dots & f(a[m][n]) \end{pmatrix}} \Big\downarrow \circledast_1$$

Definition: [md_hom]

We write

$$\texttt{md\_hom}(\; f, (\circledast_1, \dots, \circledast_d) \;)$$

for the unique d-dimensional homomorphism with combine operators $\circledast_1, \dots, \circledast_d$ and action f on singleton arrays.

PRL as MDH

PRL is an MDH — it can be expressed using the md_hom pattern (example ECR):

$\xrightarrow{\texttt{max}}$

$\texttt{weight}(\texttt{np}_1, \texttt{ep}_1)$ $\quad$ $\texttt{weight}(\texttt{np}_1, \texttt{ep}_n)$

$\texttt{weight}(\texttt{np}_m, \texttt{ep}_1)$ $\quad$ $\texttt{weight}(\texttt{np}_m, \texttt{ep}_n)$ $\downarrow +\!\!+$

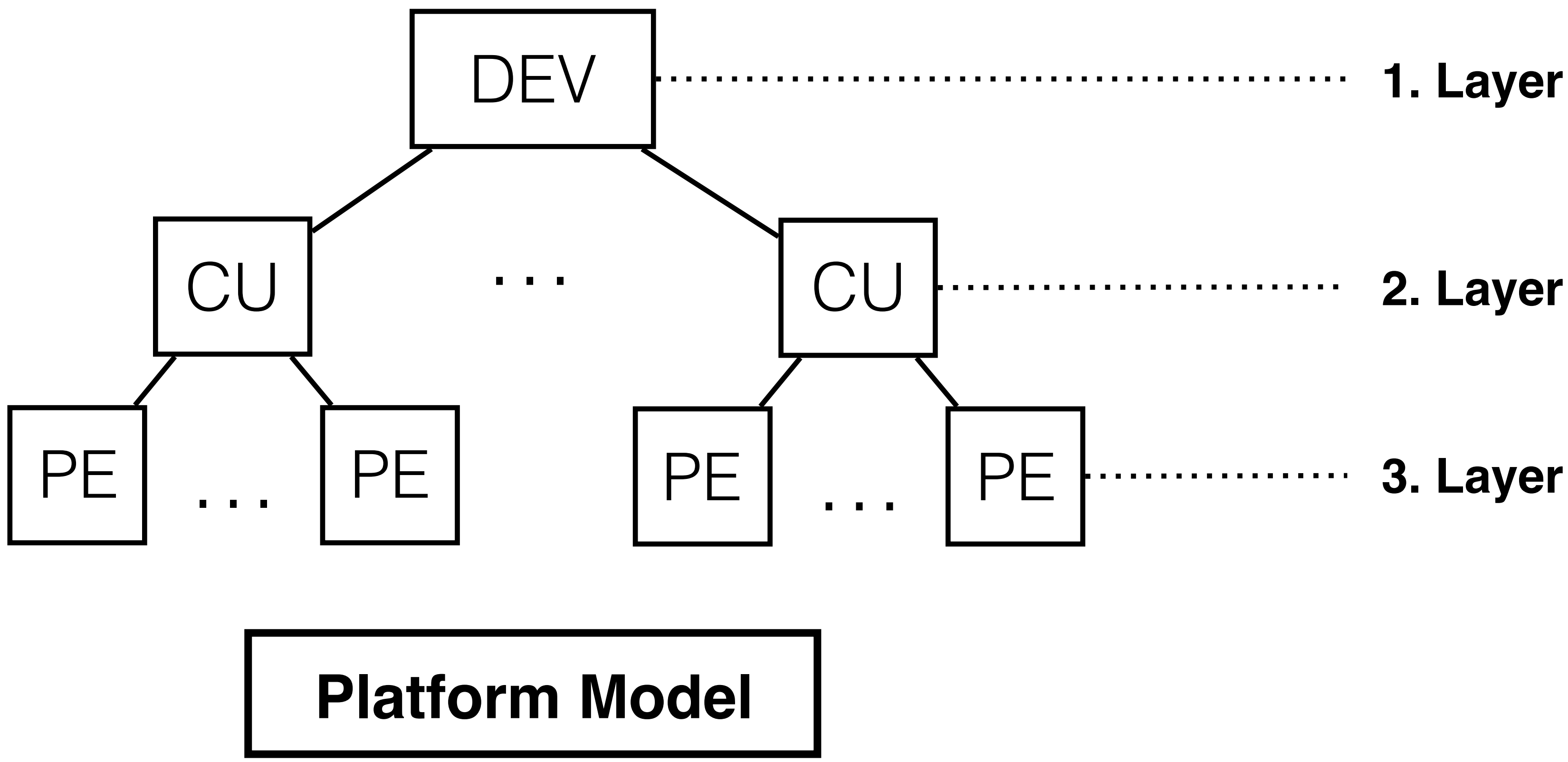
$\texttt{np}_i$: new patient $i \in \{1, \ldots, m\}$
$\texttt{ep}_i$: existing patient $i \in \{1, \ldots, n\}$

1. Build all possible pairs of new patients (1.-dim) and existing patients (2.-dim).
2. Apply function `weight` to each pair.
3. Combine results 2.-dim by operator `max` → max. matching weight for each new patient.
4. Combine results 1.-dim by operator ++ → matching weight for all existing patients

$$\text{PRL} = \texttt{md\_hom}(\ \texttt{weight}, (+\!\!+, \texttt{max})\ ) \circ \texttt{cart}$$

MDHs in OpenCL

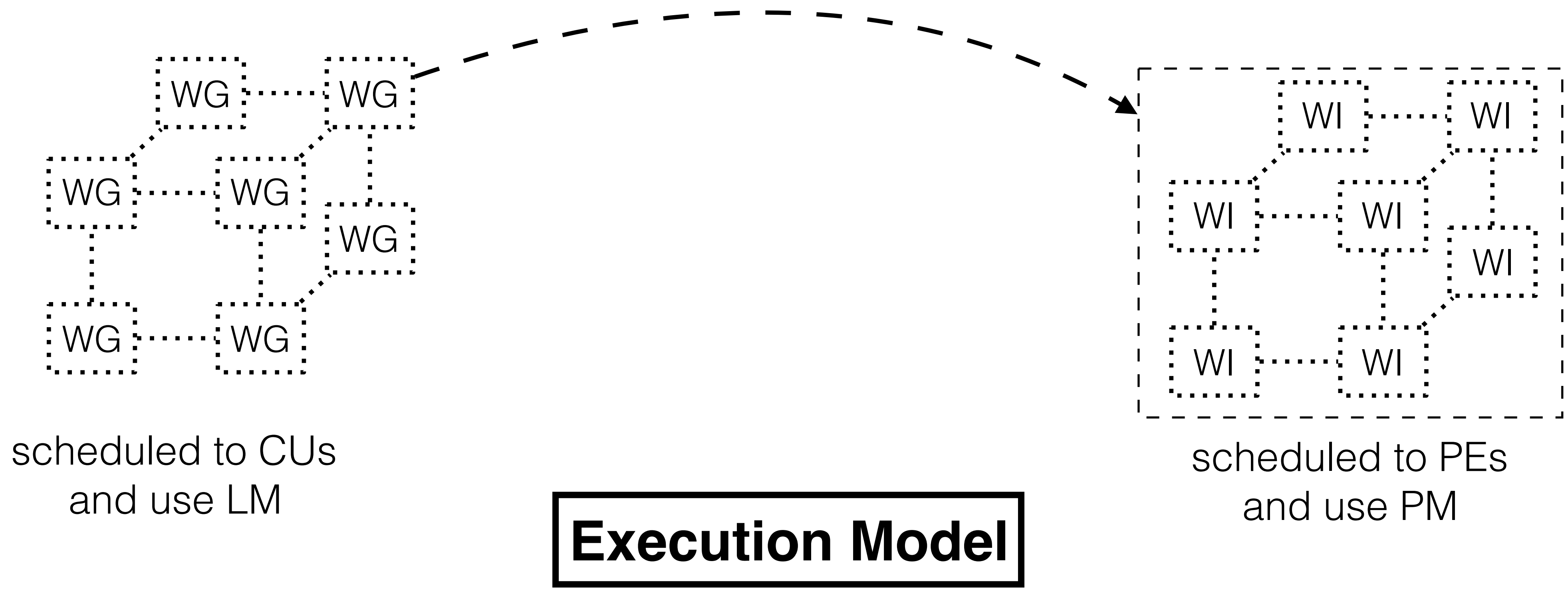
- MDHs can be efficiently implemented for CPU and GPU, e.g., in OpenCL.
- The OpenCL's models (in a nutshell):



- OpenCL has a 3-layered *Platform Model (PM)*.
- PM uniformly abstracts parallel devices (e.g., CPU or GPU).
- PM consists of *Compute Units* (e.g., cores or SMX) and *Processing Elements* (e.g., SIMD units or warps)

MDHs in OpenCL

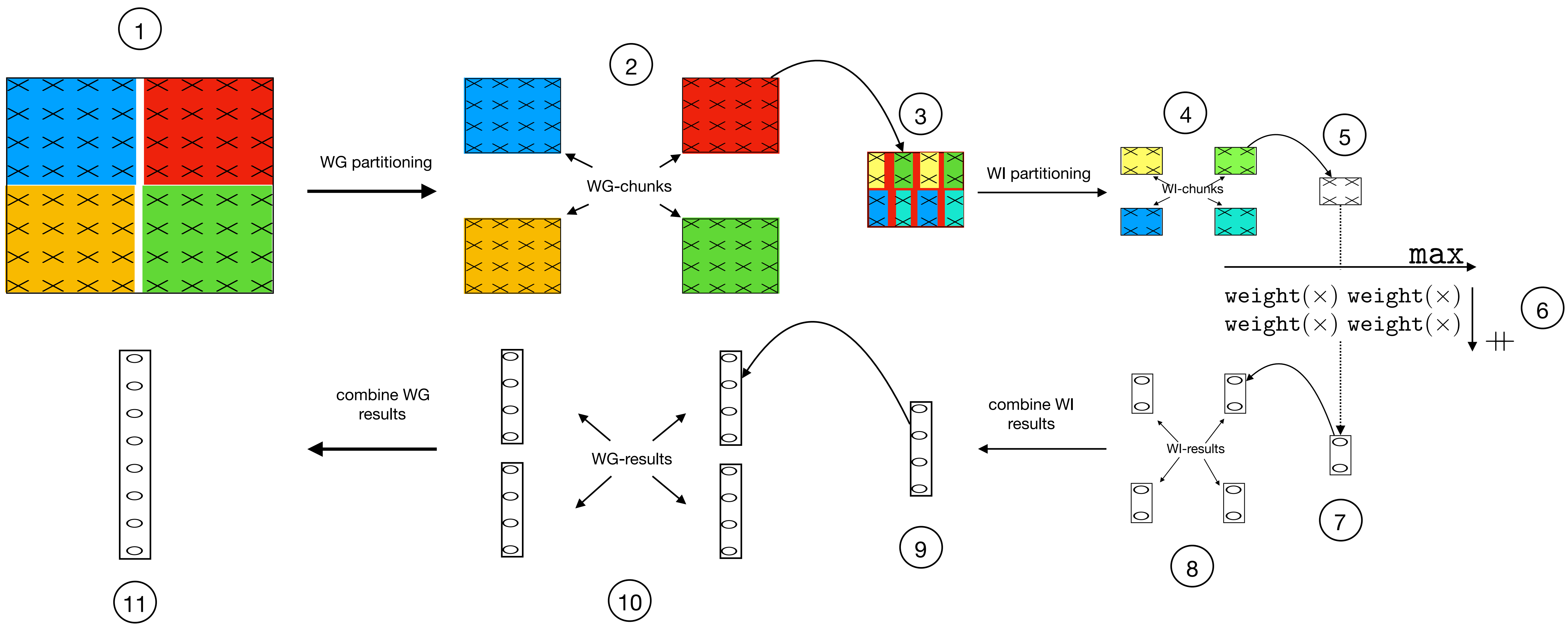
- MDHs can be efficiently implemented in OpenCL.
- The OpenCL's models (in a nutshell):



- ***Work-Groups (WG)*** scheduled to **CUs**.
- ***Work-Items (WI)*** scheduled to **PEs**.
- **Number of WGs/WIs** have to be chosen as **optimized by user** for each target combination of: i) **application**, ii) **architecture**, and iii) **input size**.

MDHs in OpenCL

The MDH OpenCL implementation schema for PRL:



We exploit the algebraic representation of PRL to split the input data for WGs and WIs..

- Our MDHs' OpenCL implementation is **generic** in the **number of WGs and WIs**.
- This **enables automatically optimizing our implementation** — for each target architecture and input size — using auto-tuning.

Automatic Performance Tuning

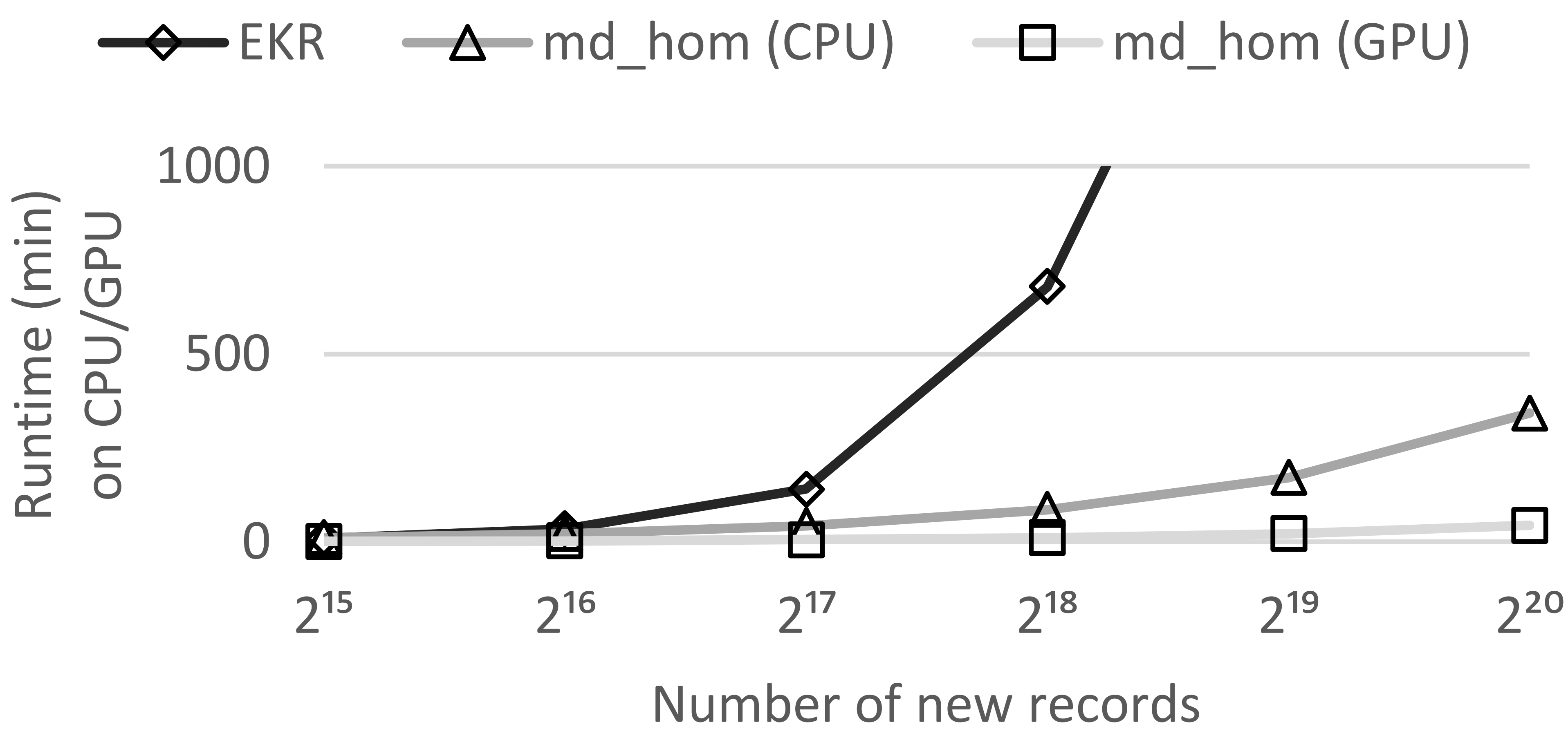
We use our ***Auto-Tuning Framework (ATF)*** to automatically chose optimized values of our performance-critical parameters.

	Domain-specific auto-tuning	OpenTuner	CLTune	ATF
Arbitrary Programming Language		✓		✓
Arbitrary Application Domain		✓	✓	✓
Arbitrary Tuning Objective	✓	✓		✓
Arbitrary Search Technique	✓	✓	✓	✓
Interdependent Parameters	✓		✓	✓
Large Parameter Ranges	✓	✓		✓
Directive-Based Auto-Tuning				✓
Automatic Cost Function Generation	✓		✓	✓

ATF combines major advantages over state-of-the-art auto-tuning approaches

[3] Rasch, Ari, and Sergei Gorlatch. "ATF: A Generic, Directive-Based Auto-Tuning Framework." *Concurrency and Computation: Practice and Experience* 31 (2019)

Experimental Results



Intel Xeon E3-1240 CPU

NVIDIA Tesla V100 GPU

- Our OpenCL implementation provides speedup **>8** on CPU as compared to ECR’s parallel Java implementation.
- This is because it can be automatically optimized (auto-tuned) for the concrete target hardware.
- Our implementation is executable also on GPUs — speedups **>80x.**

Conclusion

We present a high-performance, portable implementation of Probabilistic Record Linkage (PRL):

- Our implementation targets various parallel architectures (via OpenCL).
- It provides high performance by being automatically optimizable (via auto-tuning) for the target architecture and input size.
- Our experiments on the real-world sample of ECR show speedups of over **>8x** on Intel multi-core, and **>80x** on NVIDIA GPU.

Our approach is based on the algebraic formalism of Multi-Dimensional Homomorphisms (MDH).

Questions?