join2vec: towards efficient and semantic-rich string similarity joins

Manos Chatzakis (emmanouil.chatzakis@epfl.ch)

Supervised by Viktor Sanca and Anastasia Ailamaki

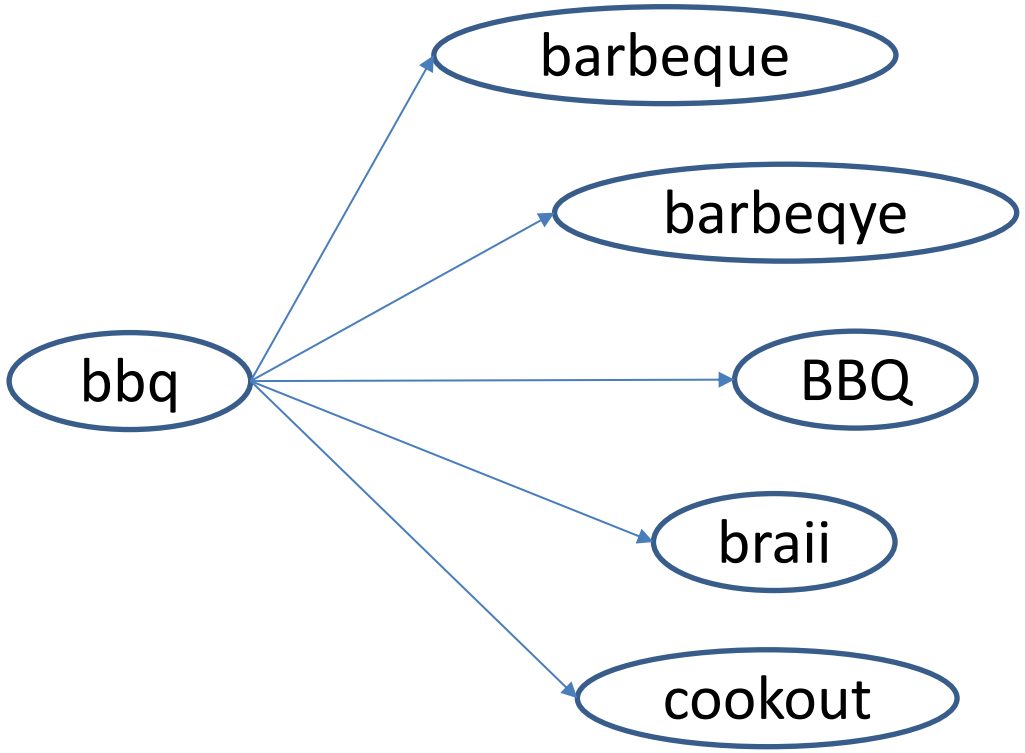
DATA-INTENSIVE APPLICATIONS AND SYSTEMS

EPFL

String Similarity Joins

...given a collection of strings, find the **most similar** pairs.

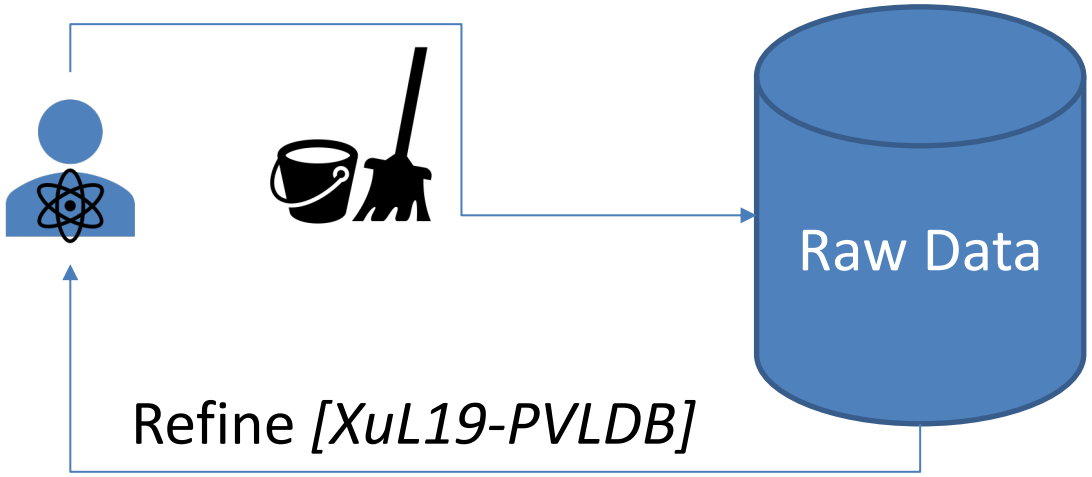
- Merge datasets by locating similar entities
- Eliminate duplicates
- Expand query terms
- Cluster and classify objects



String similarity joins are indispensable in real-world analytics

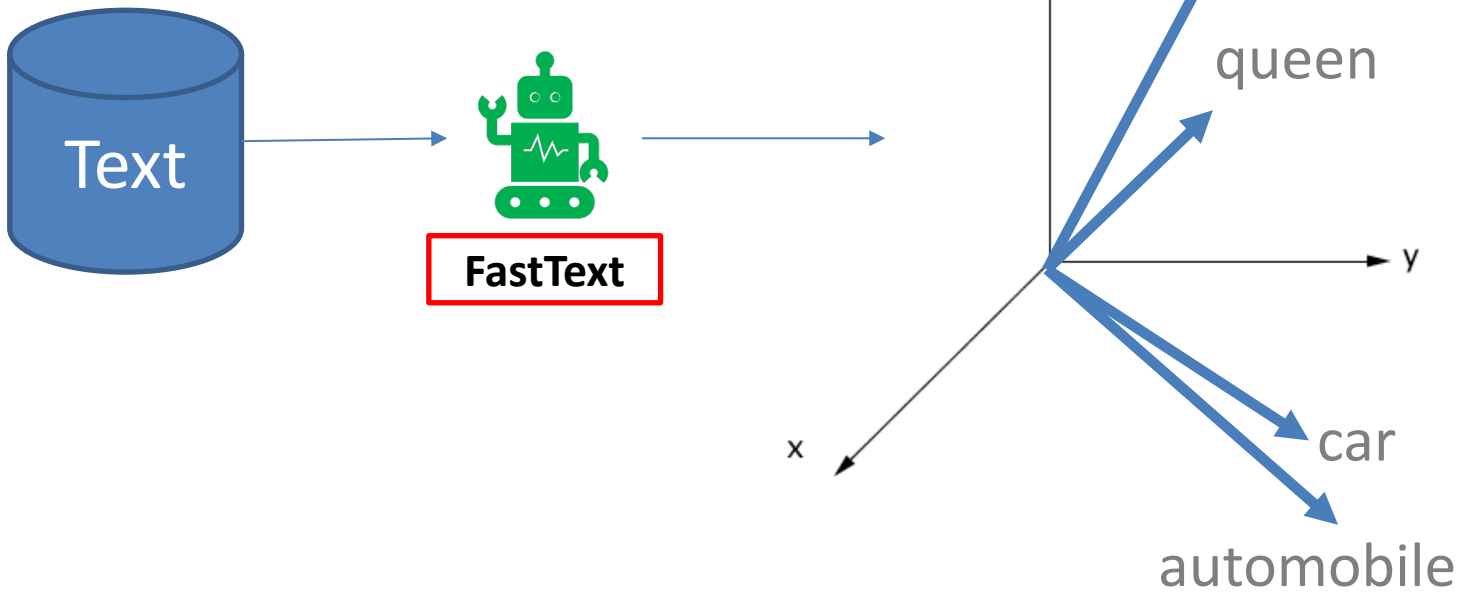
Similarity in Practice

Define similarity rules (Syntactic, Synonym, Taxonomy)



Defining similarity rules for strings is a difficult task

Capturing String Context

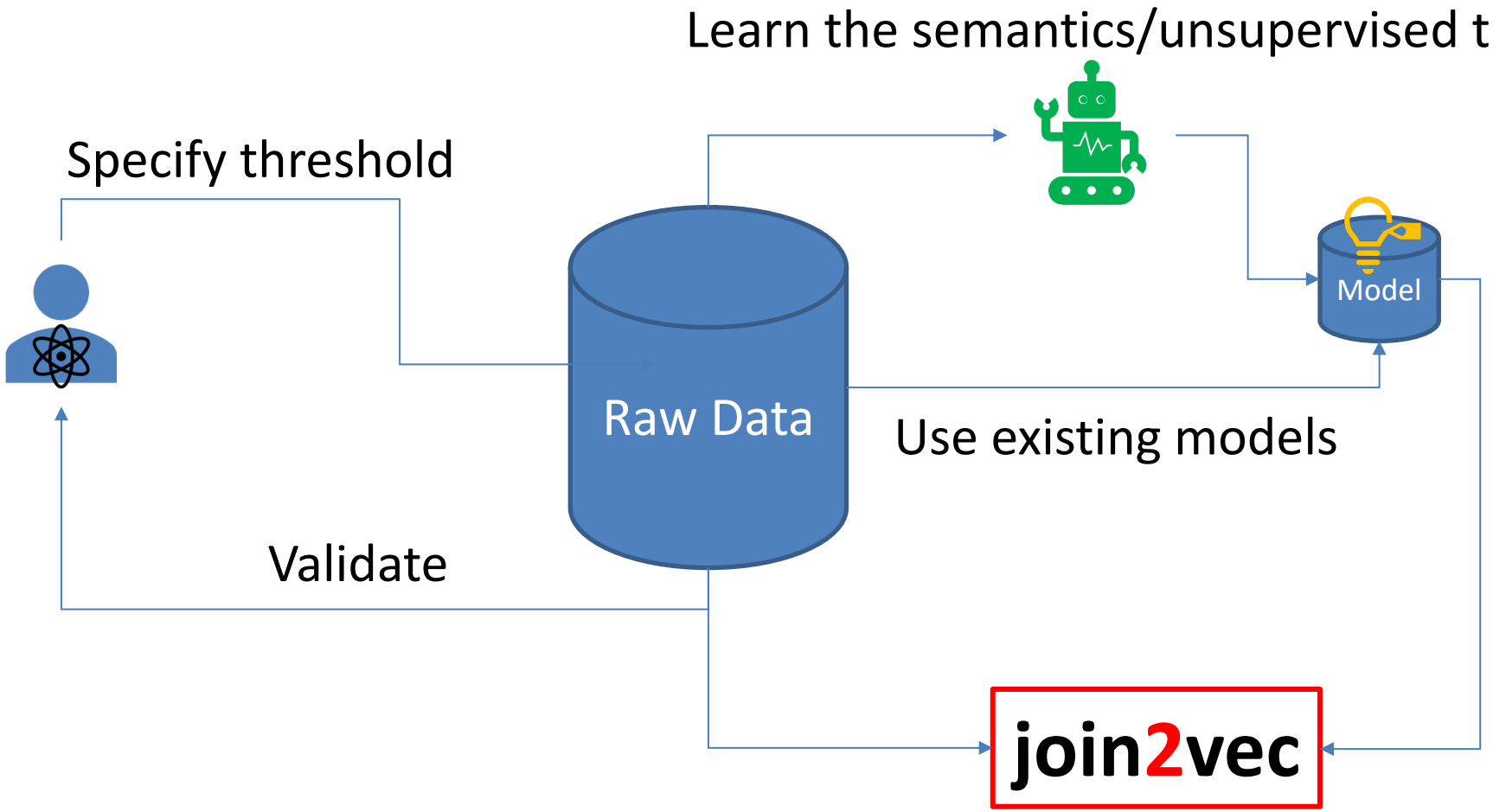


Embeddings are high dimensional vectors

Automated similarity metric

Unsupervised learning enables capturing both semantic and domain-specific context

join2vec



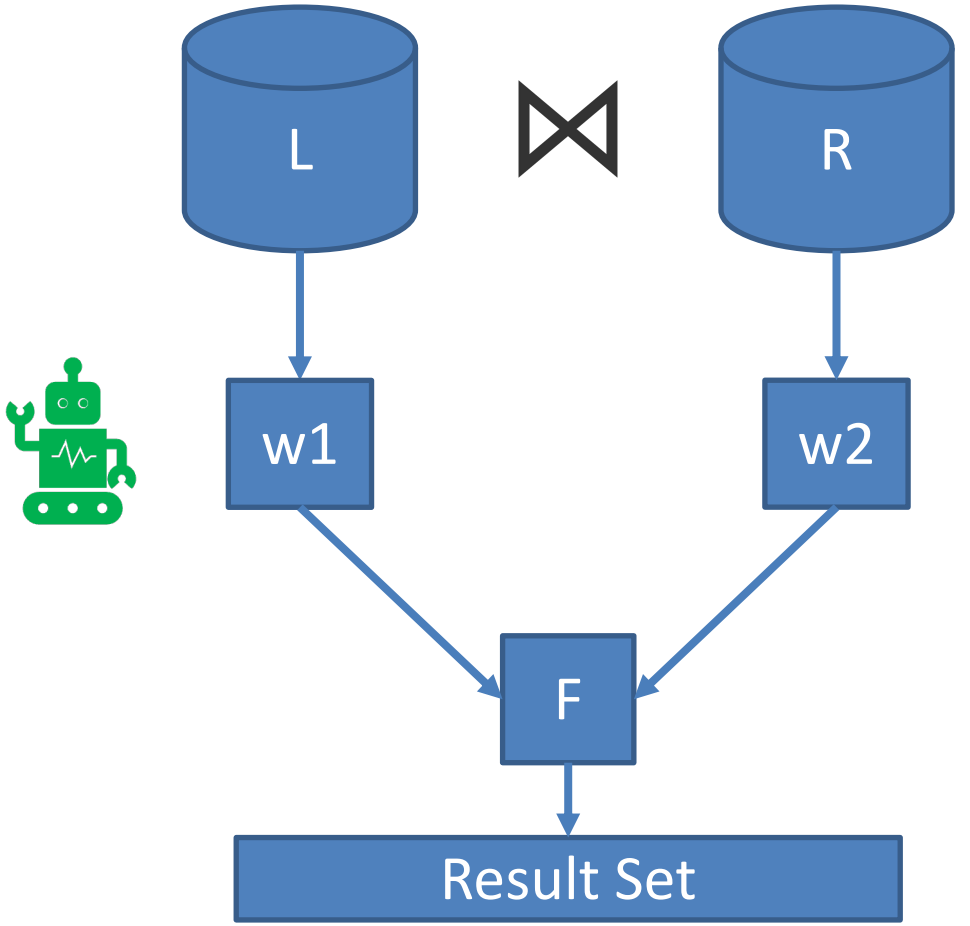
Shift similarity specification from user- to model-driven, automated rules

Automated Context-Rich Similarity

S	R	Score	Type
Bbq	Barbeque	0.90	Synonym
Necklaces	Earrings	0.91	Taxonomy
Sydney	Melbourne	0.95	
Burritos	Tacos	0.90	
Gynaecologist	Gynecologist	0.91	Syntactic
Syllables	Syllable	0.96	

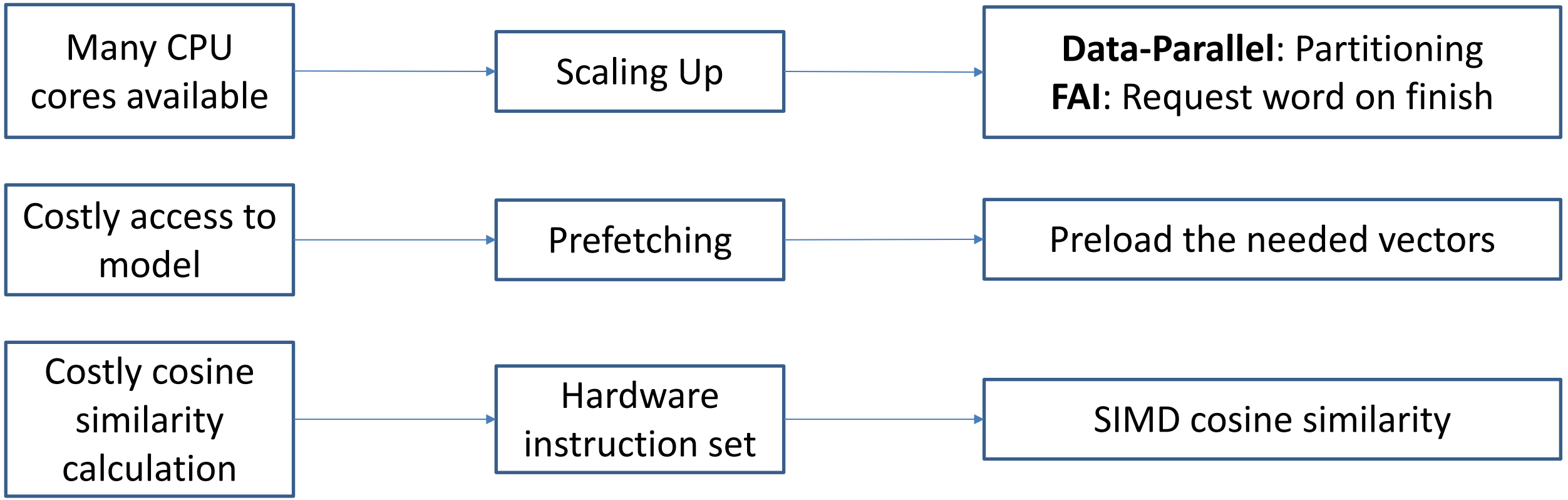
Embedding models capture different types of string similarity

Join2vec Algorithm



Goal: Tight join-model integration for efficient execution

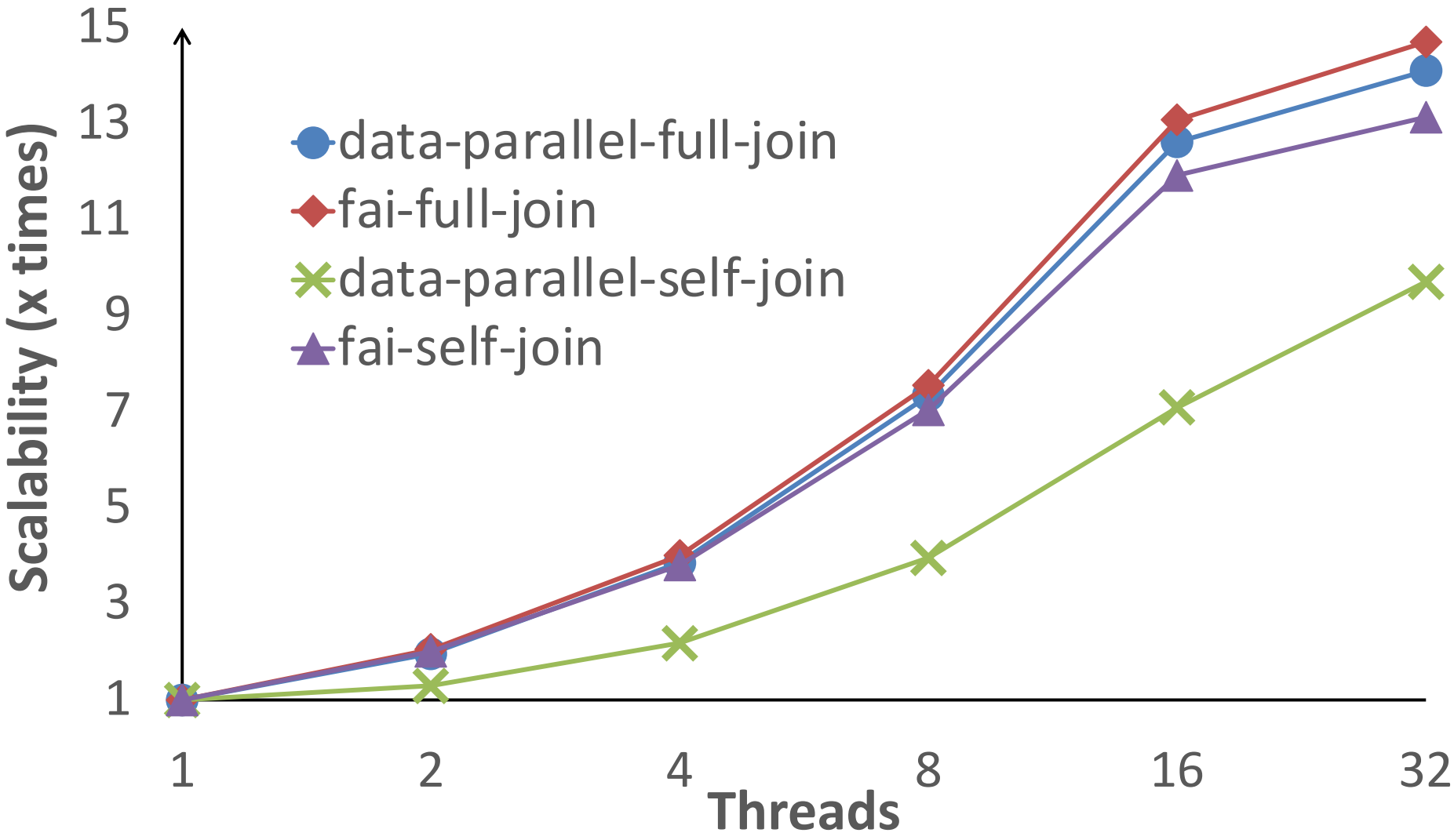
Hardware Concious Optimizations



Experimental Setup

- **Diascld25** server
  - 2 x Intel Xeon E5-2660 (2.2GHz, 8 cores, 2 threads per core)
  - 126GB memory
  - Ubuntu 18.04.4 LTS, C/C++, version g++ 7.5.0
- Dataset: **Wikipedia** words
  - 100K Randomly selected
    - No stopwords/duplicates
  - 7.5 average length
- Model
  - **FastText** trained on full Wikipedia dataset

Scaling up the algorithm



Threshold=0.9

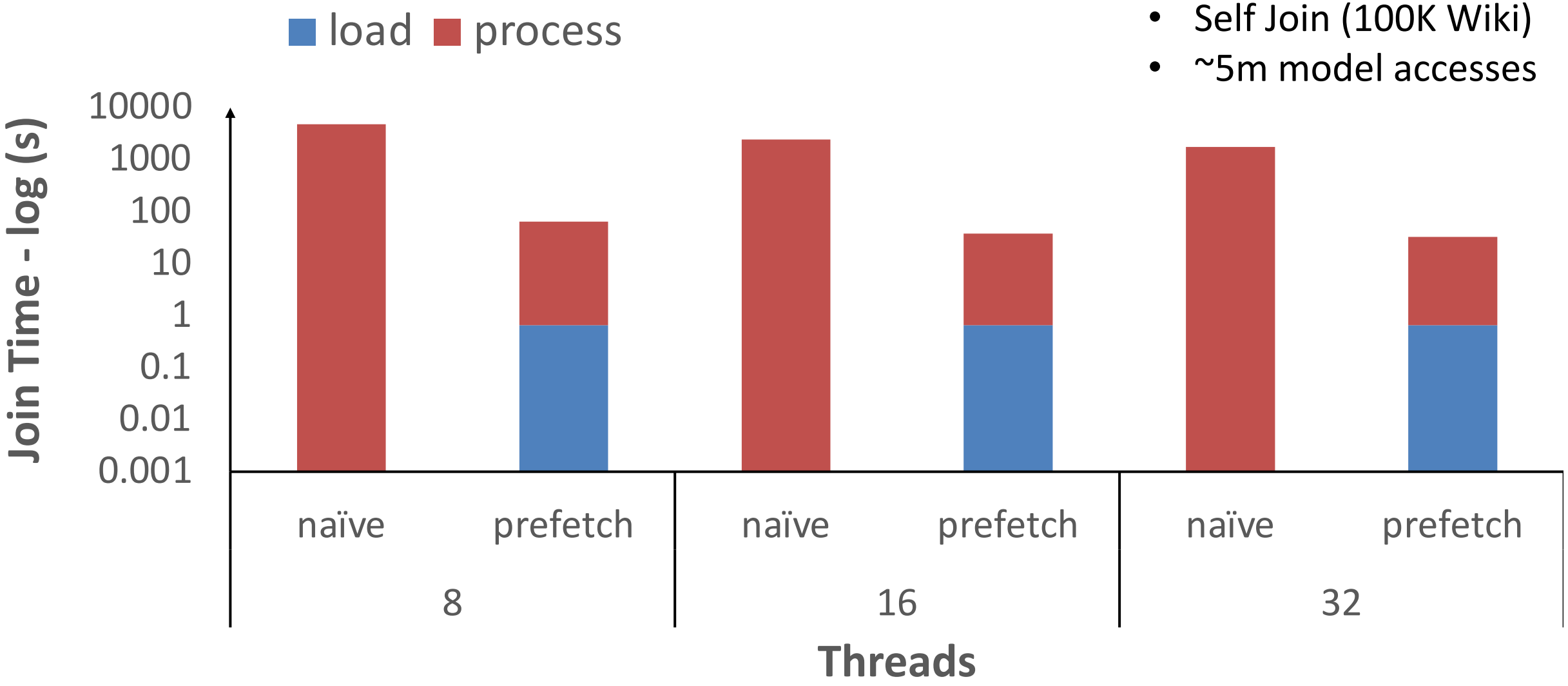
Self Join
#Input=100K Wiki
#Output=32763 tuples

Full Join
#Input=100K*100K Wiki
#Output=144433 tuples

Both methods scale the same for full joins, FAI scales better for Self Join

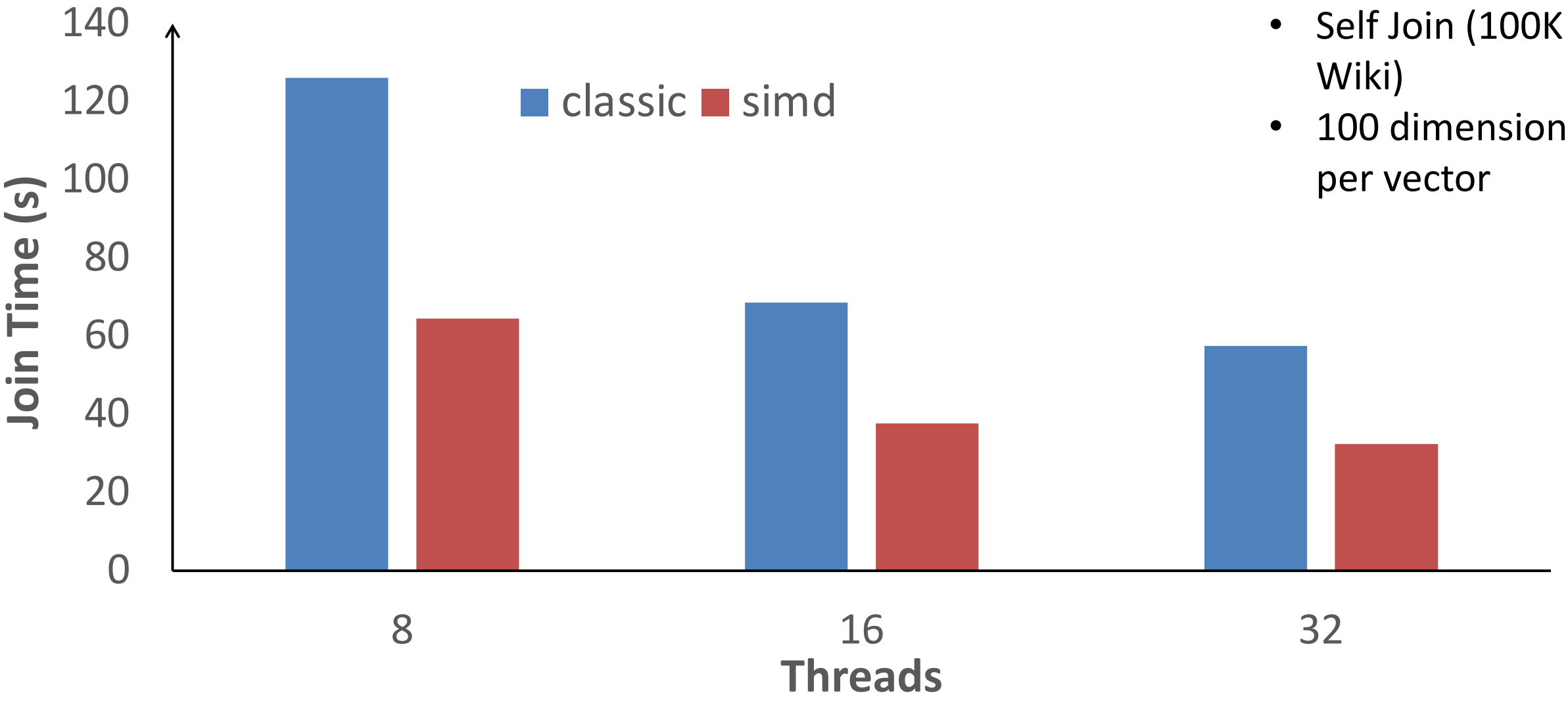
Optimizing the Data Access Pattern

- Self Join (100K Wiki)
- ~5m model accesses



Fetching then executing is 30X faster than mixed data access

Optimizing Frequent Operations



- Self Join (100K Wiki)
- 100 dimensions per vector

SIMD cosine similarity provides 1.8X better execution time

Efficient and Semantic Rich SimJoin

- Integrating state of the art ML model with similarity join
- Explore hardware optimizations for scalable execution
- 48x faster execution, compared to the naively scaled-up solution

Thank you!