ETH zürich

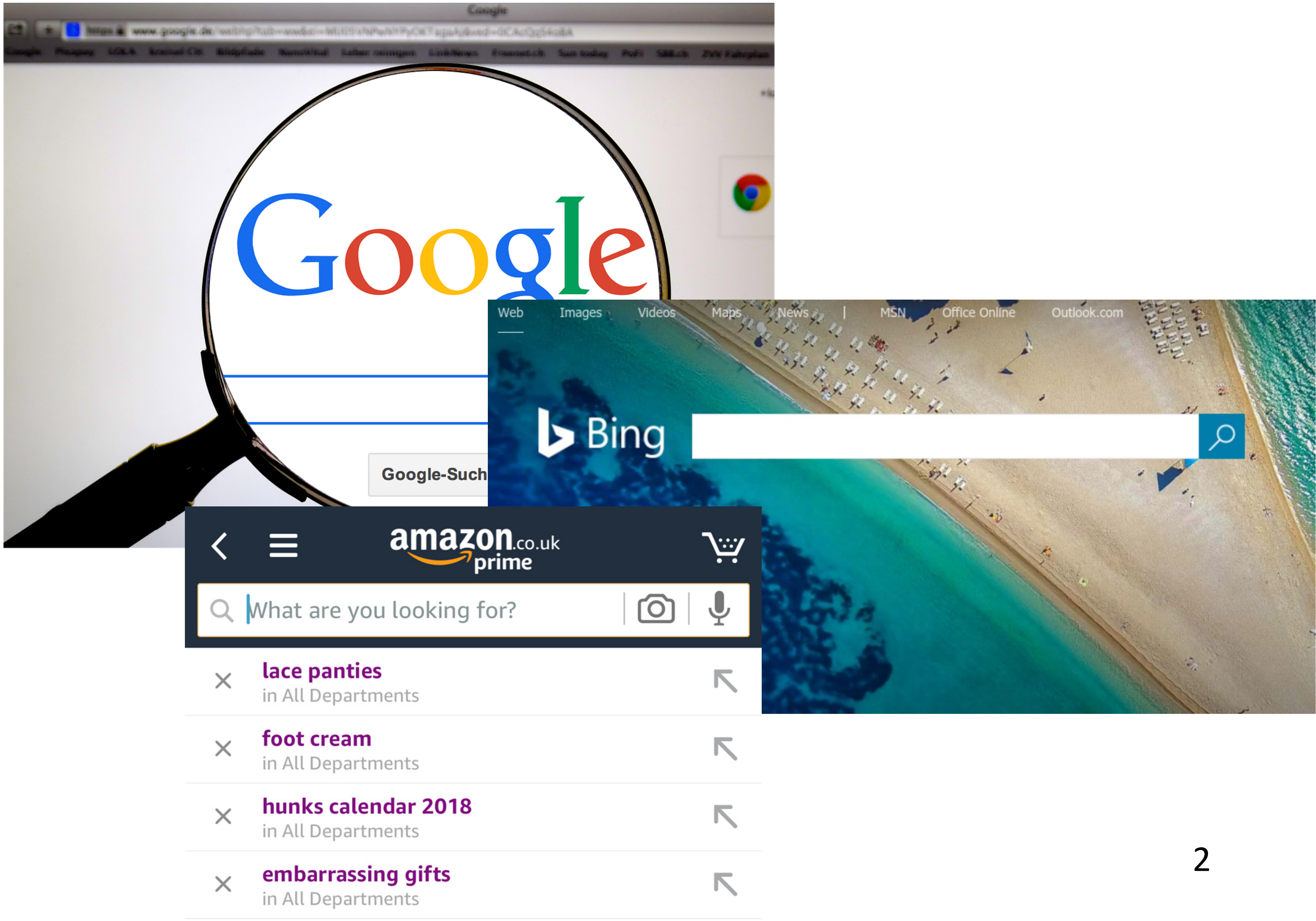
Co-design Hardware and Algorithm for Vector Search

Wenqi Jiang

Systems Group, Dept. of Computer Science, ETH Zürich

2023/06/23

Systems@ETH zürich



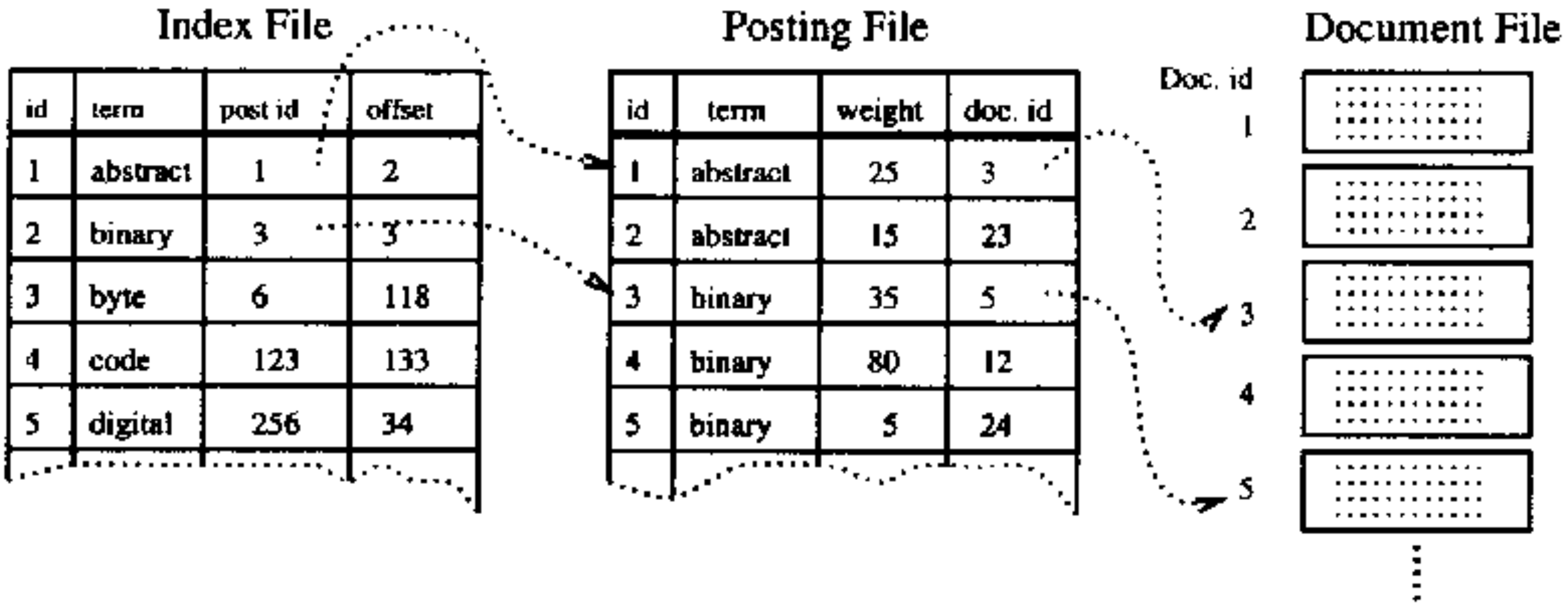
lace panties
in All Departments

foot cream
in All Departments

hunks calendar 2018
in All Departments

embarrassing gifts
in All Departments

Keyword-based matching was the theme of search engines for decades



Index File

id	term	post id	offset
1	abstract	1	2
2	binary	3	3
3	byte	6	118
4	code	123	133
5	digital	256	34

Posting File

id	term	weight	doc. id
1	abstract	25	3
2	abstract	15	23
3	binary	35	5
4	binary	80	12
5	binary	5	24

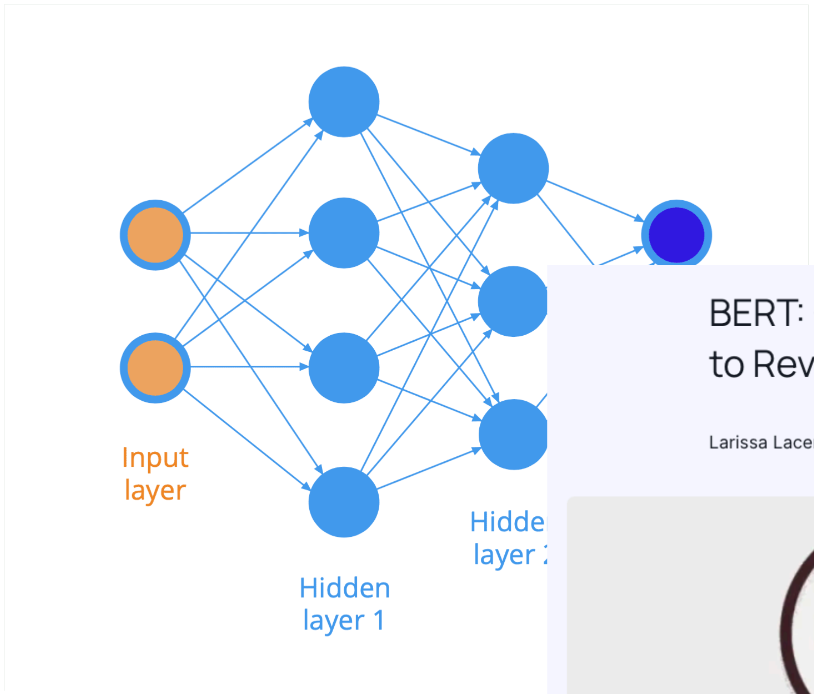
What’s wrong with this method?

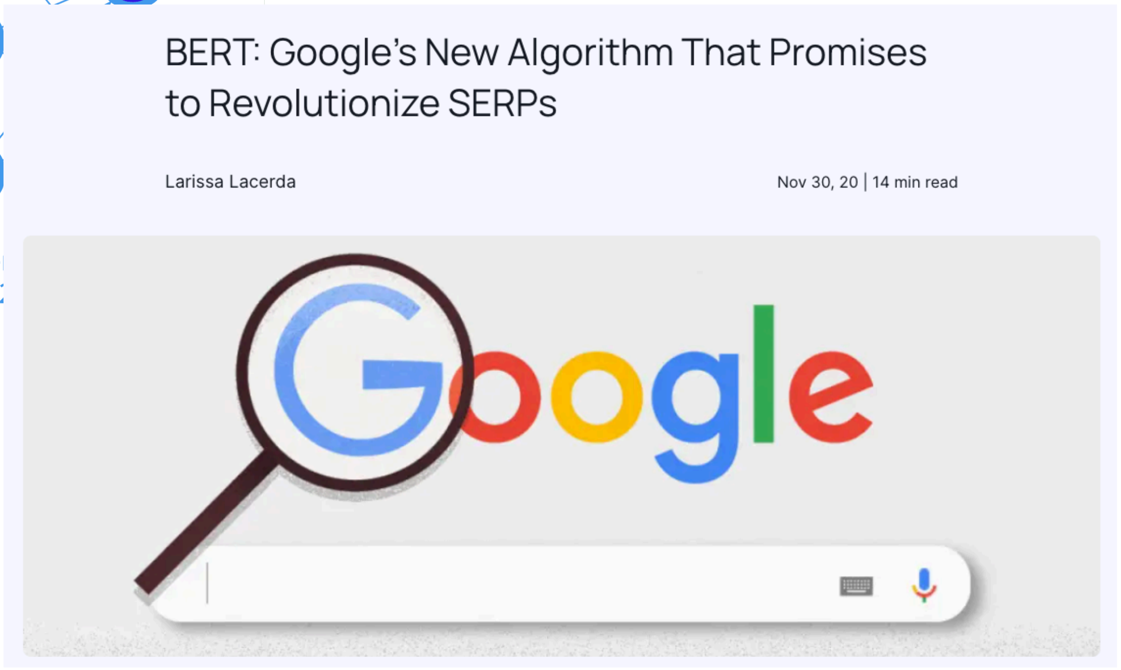
Natural languages are complex: two sentences can share the same meaning while sharing little common words

We’re looking for a “tragic love story” but Shakespeare wrote about “star-crossed lovers”

Exact match is powerless in this case…

Machine learning drives the new generation of search engines



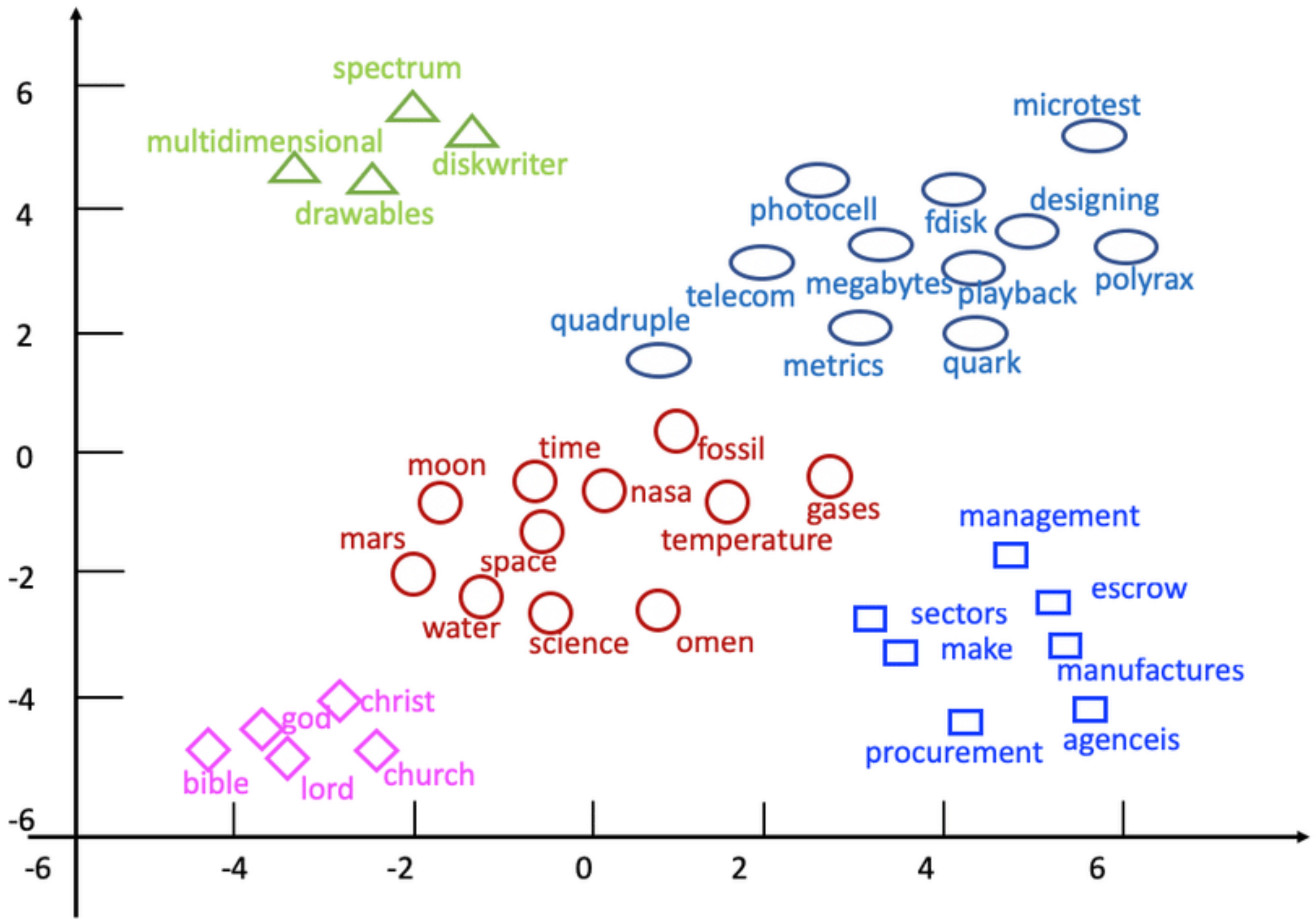

BERT: Google's New Algorithm That Promises to Revolutionize SERPs

Larissa Lacerda

Nov 30, 20 | 14 min read

Words to vectors

Similar concept in document encoding: doc2vec



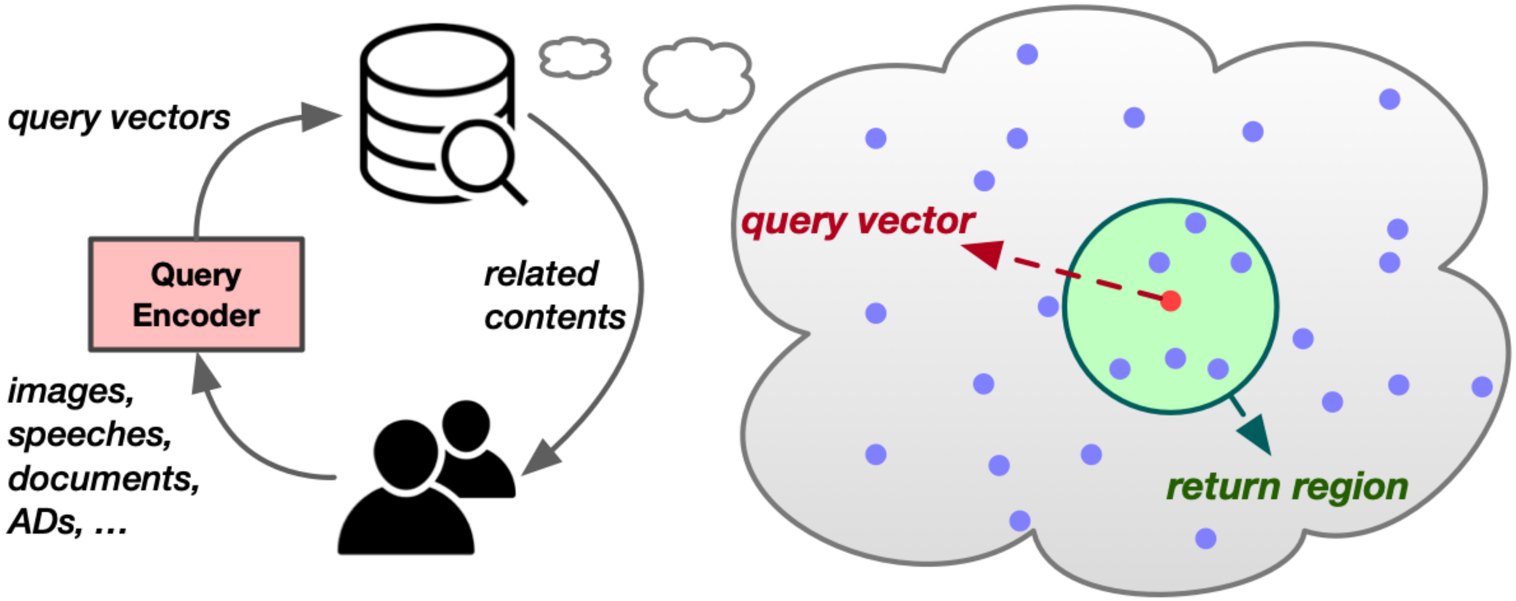
Approximate nearest neighbor search (ANNS)

Given: a set of database vectors (e.g., encoded documents)

Input: a query vector

Output: K most similar vectors in the database

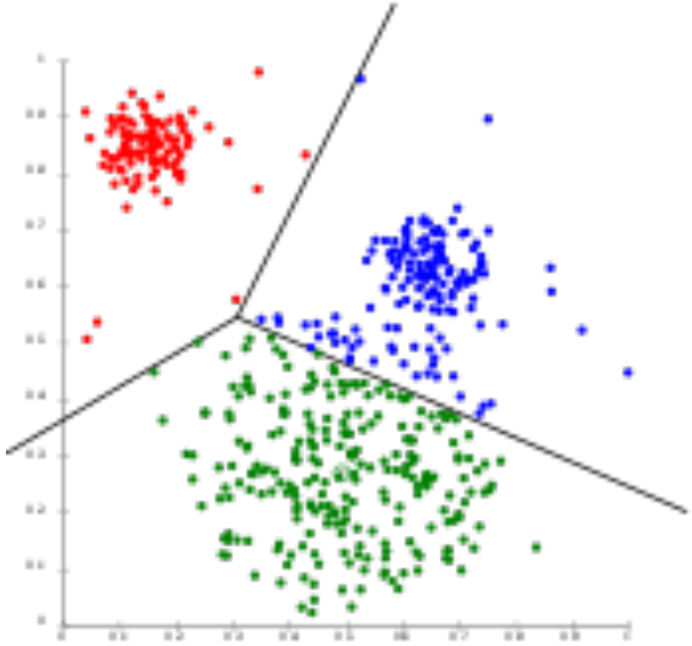
Quality metric: recall



Target algorithm: IVF-PQ

IVF: inverted file index

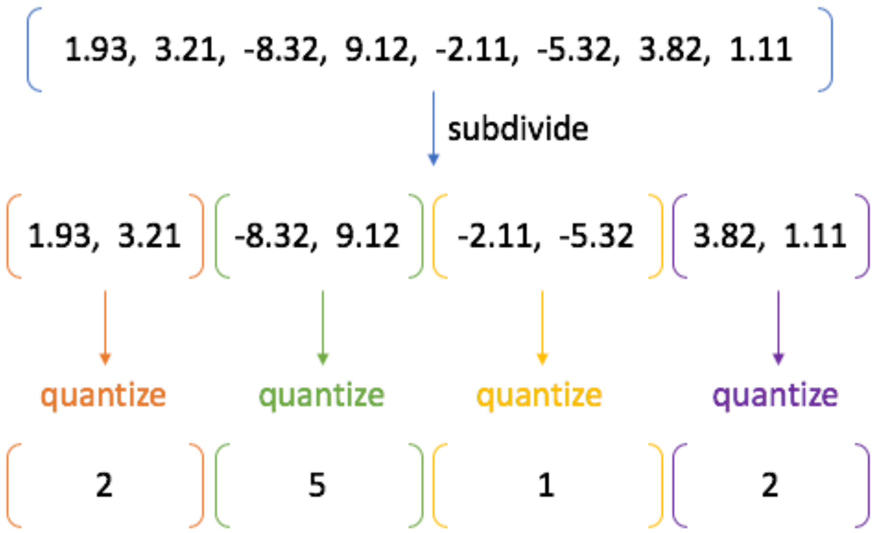
- Not the keyword based one!
- Partition the vectors by clustering

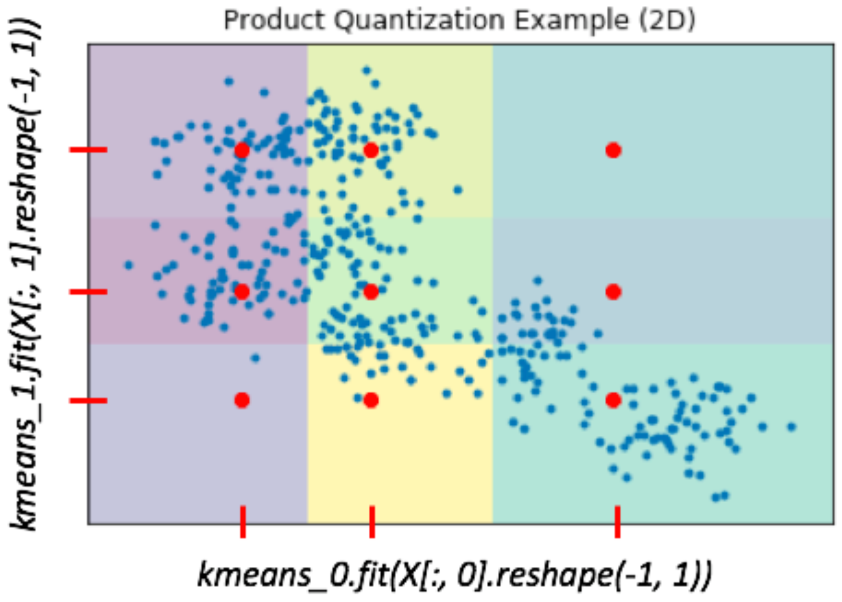


Target algorithm: IVF-PQ

PQ: product quantization

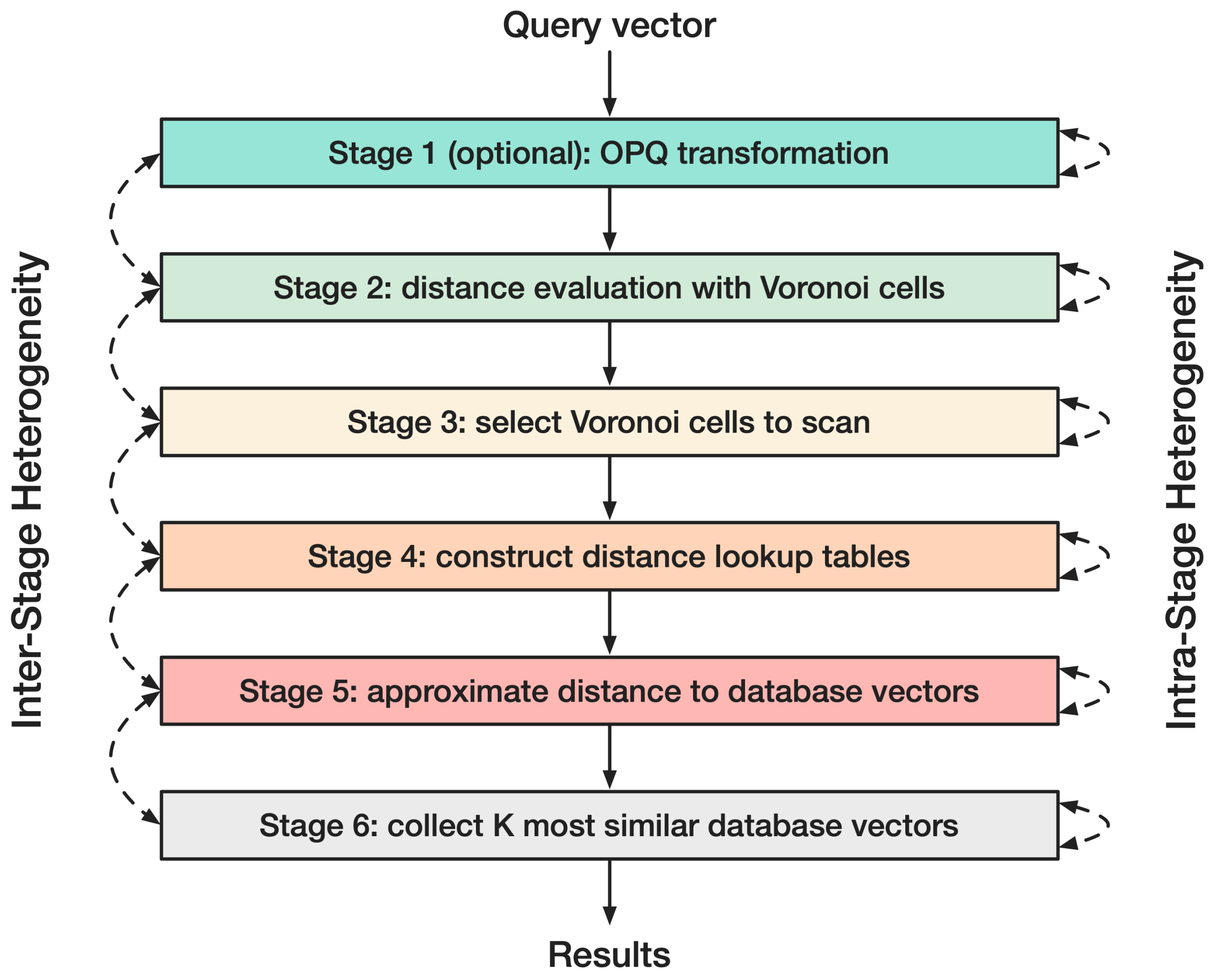
- Reduce the vector size to a few bytes
- Allow fully in-memory search





$X = (x_0, x_1)$

IVF-PQ: search process


Query vector
Stage 1 (optional): OPQ transformation
Stage 2: distance evaluation with Voronoi cells
Stage 3: select Voronoi cells to scan
Stage 4: construct distance lookup tables
Stage 5: approximate distance to database vectors
Stage 6: collect K most similar database vectors
Results
Inter-Stage Heterogeneity
Intra-Stage Heterogeneity

Can we put the algorithm on specialized hardware?

To build a hardware accelerator for the algorithm, we need to identify the bottleneck first

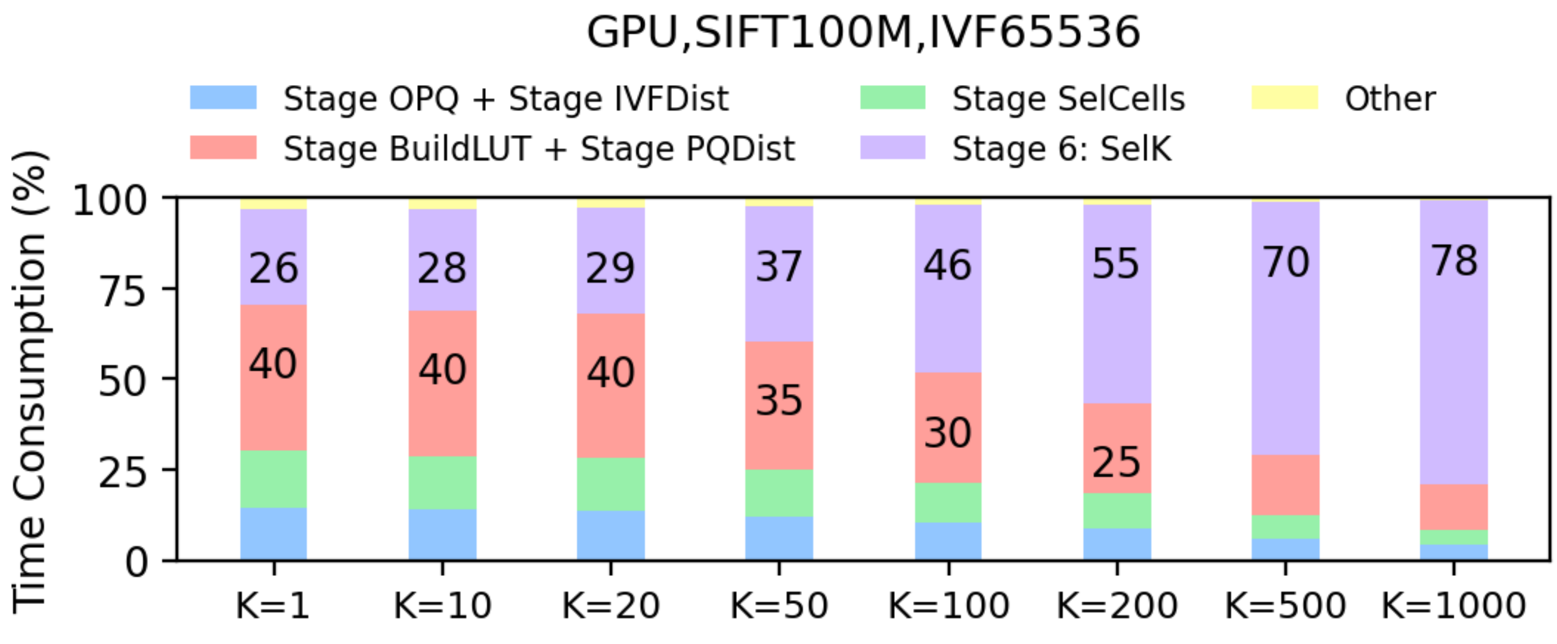
However, there are many parameters in IVF-PQ…

- *nlist*: the number of clusters (partitions) in the index
- *nprobe*: the number of clusters to visit per search
- *K*: the number of results to return per query
- …

These influence the bottleneck dramatically!

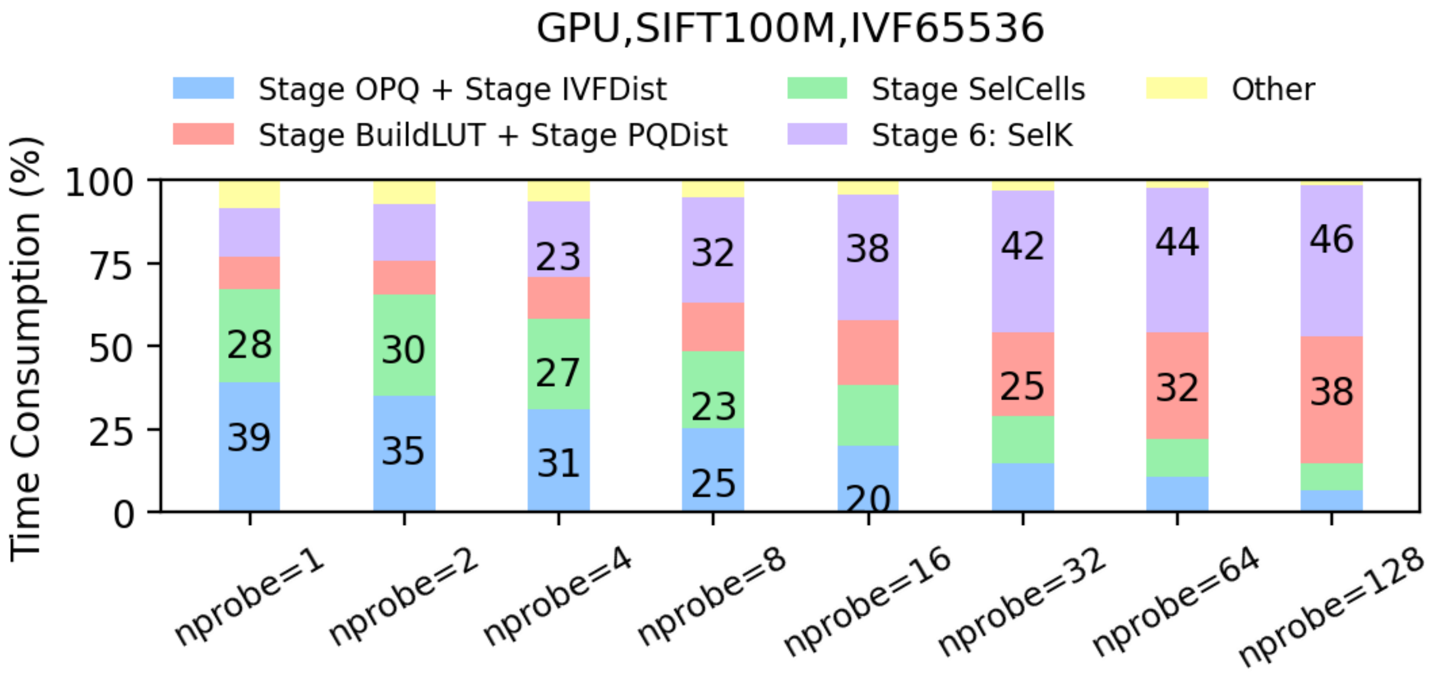
The effect of K on performance bottlenecks

K = number of results to return



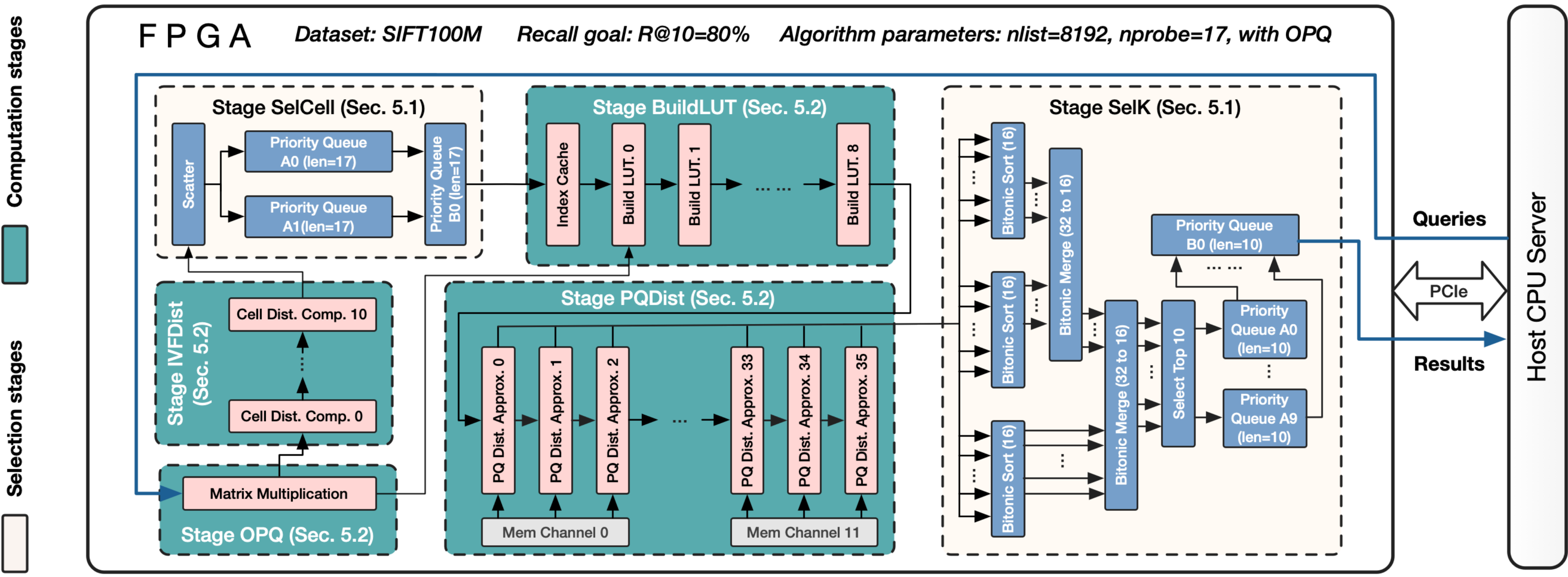
The effect of *nprobe* on performance bottlenecks

nprobe = number of clusters to scan



Example FPGA design

Problem: the resource utilization (number of processing elements) per stage by human expert, which can be suboptimal



Goal in real industry deployments

A fixed recall goal requirements in search engines

Can we build an FPGA-based ANNS systems that:

Given:

(a) A user-provided dataset

(b) A recall requirement

Figure out:

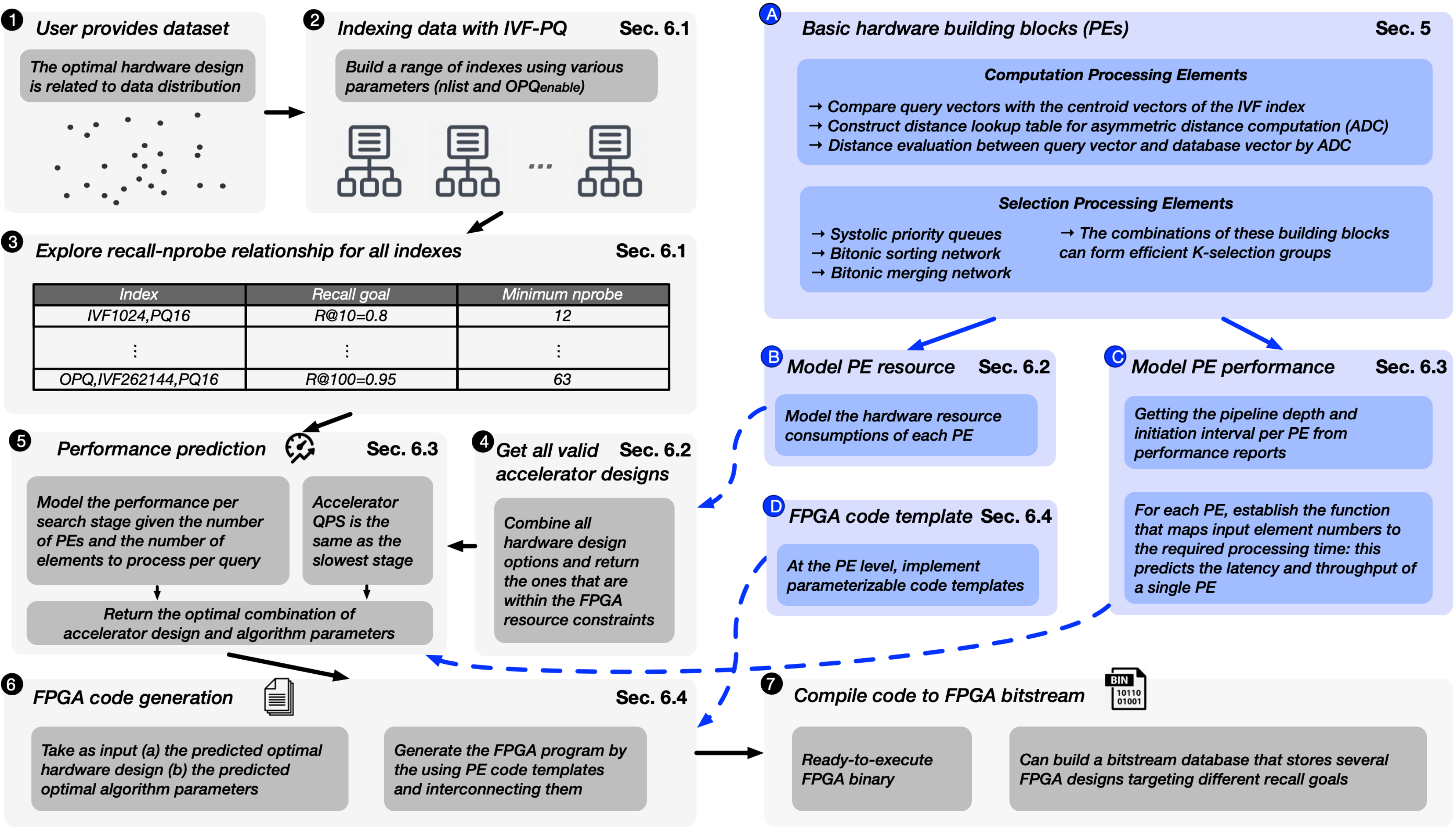
(a) The best accelerator design

(b) The according algorithm parameters to use

Generate:

The optimal hardware accelerator customized for the optimal algorithm parameters

FANNS: co-design hardware and algorithm for vector search



❶ *User provides dataset*

The optimal hardware design is related to data distribution

❷ *Indexing data with IVF-PQ* **Sec. 6.1**

Build a range of indexes using various parameters (nlist and OPQ$_{enable}$)

❸ *Explore recall-nprobe relationship for all indexes* **Sec. 6.1**

Index	Recall goal	Minimum nprobe
IVF1024,PQ16	R@10=0.8	12
⋮	⋮	⋮
OPQ,IVF262144,PQ16	R@100=0.95	63

❹ *Get all valid accelerator designs* **Sec. 6.2**

Combine all hardware design options and return the ones that are within the FPGA resource constraints

❺ *Performance prediction* **Sec. 6.3**

Model the performance per search stage given the number of PEs and the number of elements to process per query

Accelerator QPS is the same as the slowest stage

Return the optimal combination of accelerator design and algorithm parameters

❻ *FPGA code generation* **Sec. 6.4**

Take as input (a) the predicted optimal hardware design (b) the predicted optimal algorithm parameters

Generate the FPGA program by the using PE code templates and interconnecting them

❼ *Compile code to FPGA bitstream*

Ready-to-execute FPGA binary

Can build a bitstream database that stores several FPGA designs targeting different recall goals

Ⓐ *Basic hardware building blocks (PEs)* **Sec. 5**

Computation Processing Elements

- *→ Compare query vectors with the centroid vectors of the IVF index*
- *→ Construct distance lookup table for asymmetric distance computation (ADC)*
- *→ Distance evaluation between query vector and database vector by ADC*

Selection Processing Elements

- *→ Systolic priority queues*
- *→ Bitonic sorting network*
- *→ Bitonic merging network*

→ The combinations of these building blocks can form efficient K-selection groups

Ⓑ *Model PE resource* **Sec. 6.2**

Model the hardware resource consumptions of each PE

Ⓒ *Model PE performance* **Sec. 6.3**

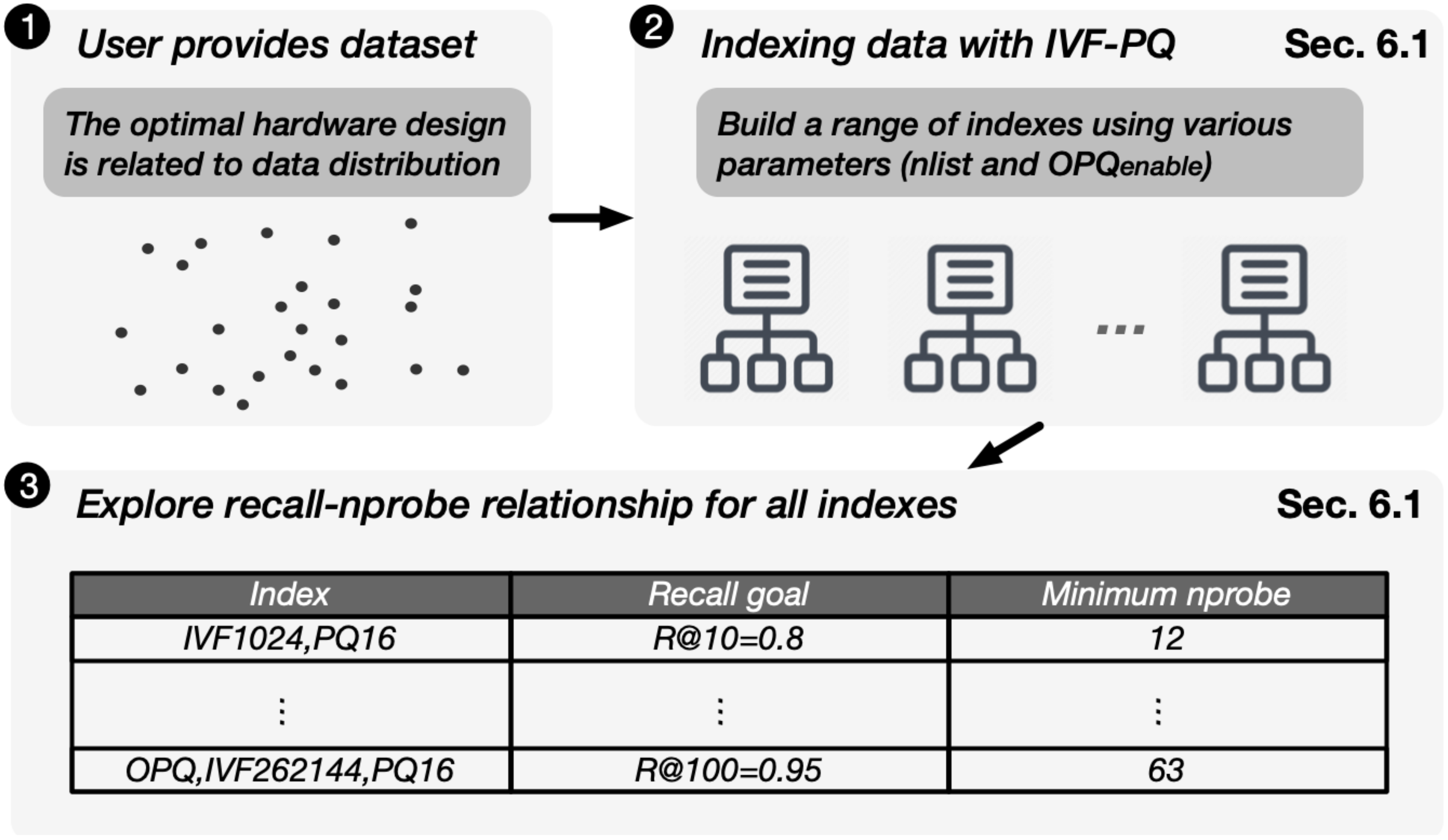
Getting the pipeline depth and initiation interval per PE from performance reports

For each PE, establish the function that maps input element numbers to the required processing time: this predicts the latency and throughput of a single PE

Ⓓ *FPGA code template* **Sec. 6.4**

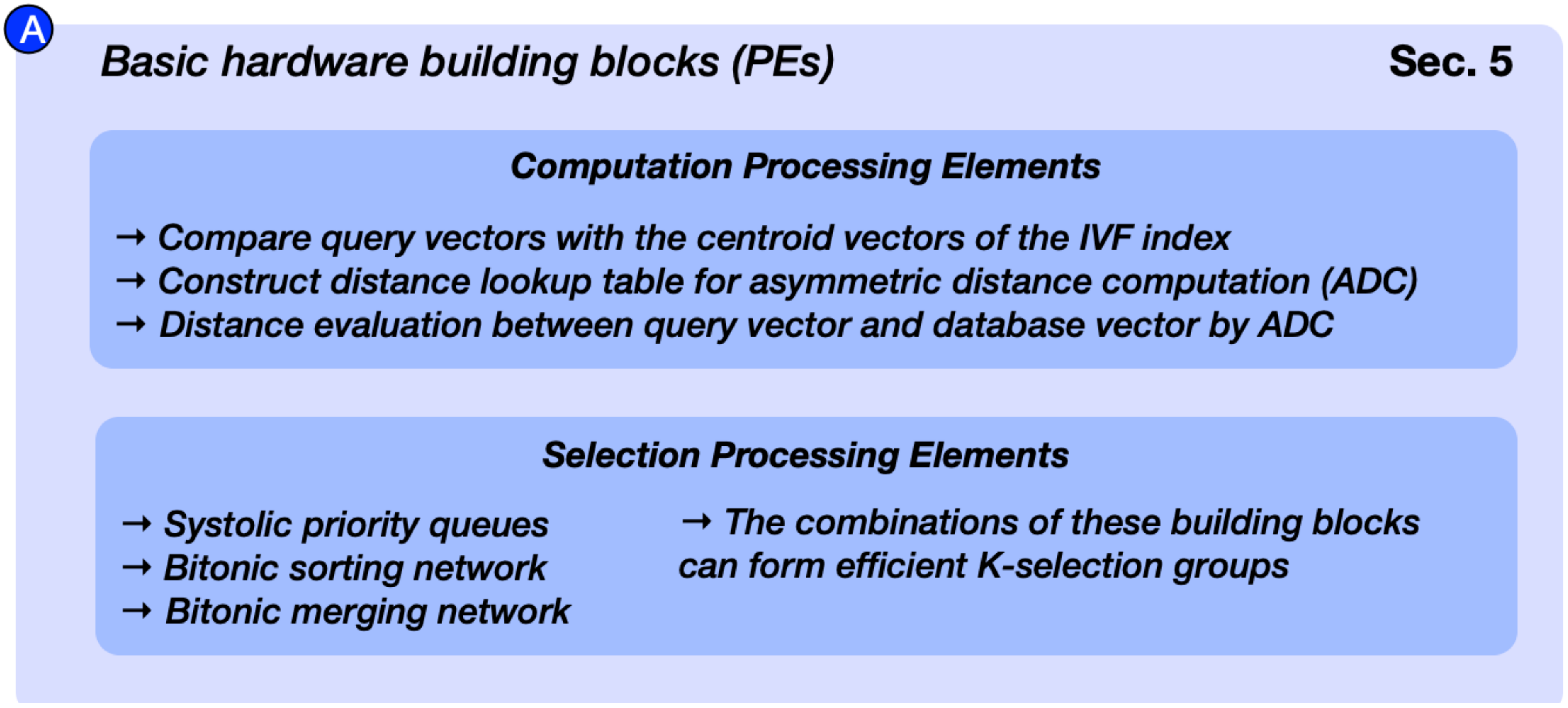
At the PE level, implement parameterizable code templates

On the algorithm side


1 User provides dataset
The optimal hardware design is related to data distribution
2 Indexing data with IVF-PQ
Sec. 6.1
Build a range of indexes using various parameters (nlist and OPQ$_{enable}$)
...
3 Explore recall-nprobe relationship for all indexes
Sec. 6.1

Index	Recall goal	Minimum nprobe
IVF1024,PQ16	R@10=0.8	12
⋮	⋮	⋮
OPQ,IVF262144,PQ16	R@100=0.95	63

On the hardware side



A

Basic hardware building blocks (PEs) **Sec. 5**

Computation Processing Elements

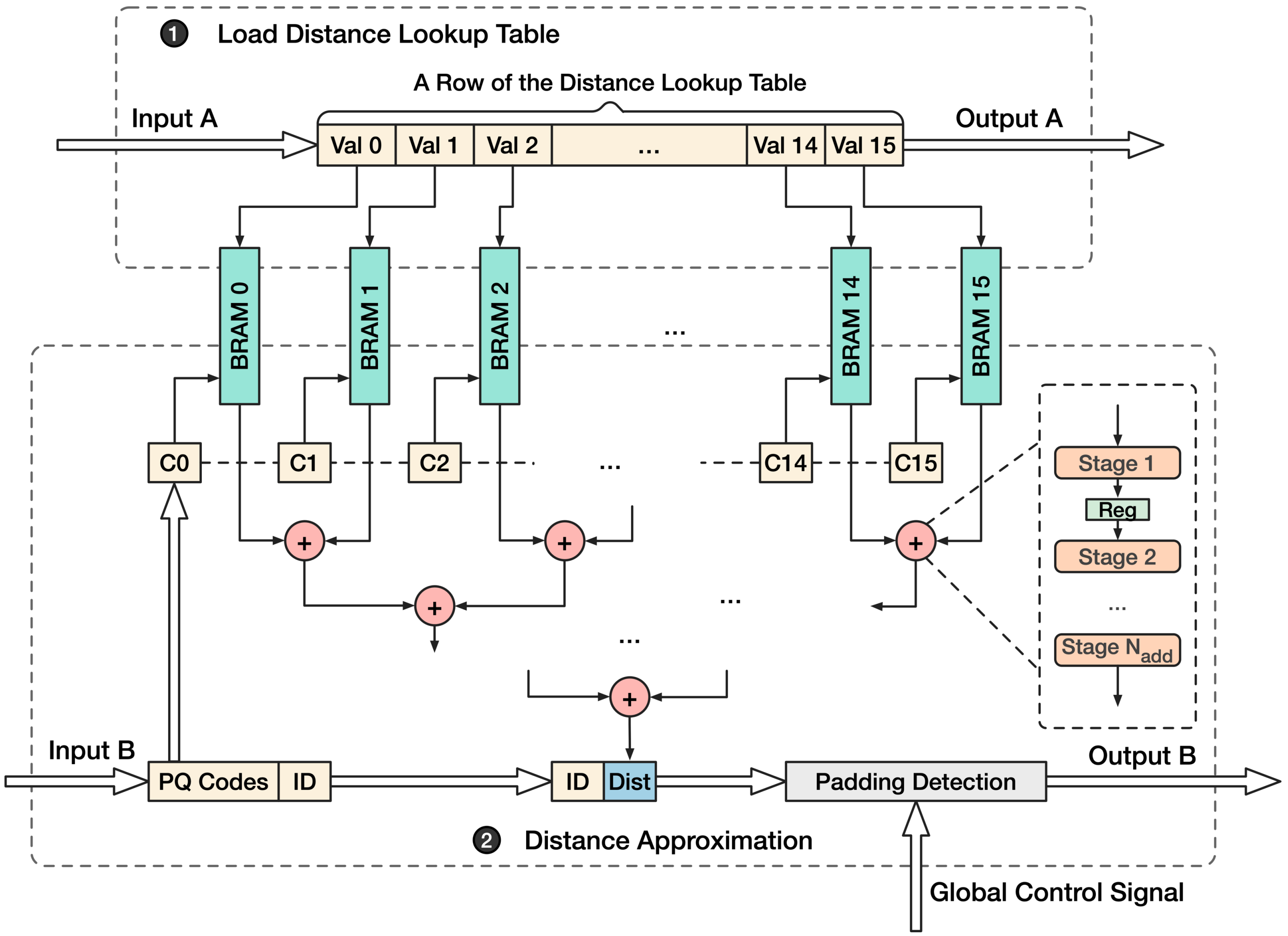
→ *Compare query vectors with the centroid vectors of the IVF index*
→ *Construct distance lookup table for asymmetric distance computation (ADC)*
→ *Distance evaluation between query vector and database vector by ADC*

Selection Processing Elements

→ *Systolic priority queues*
→ *Bitonic sorting network*
→ *Bitonic merging network*

→ *The combinations of these building blocks can form efficient K-selection groups*

Hardware design: distance estimation



On the hardware side

B ***Model PE resource*** **Sec. 6.2**

Model the hardware resource consumptions of each PE

C ***Model PE performance*** **Sec. 6.3**

Getting the pipeline depth and initiation interval per PE from performance reports

For each PE, establish the function that maps input element numbers to the required processing time: this predicts the latency and throughput of a single PE

D ***FPGA code template*** **Sec. 6.4**

At the PE level, implement parameterizable code templates

Valid accelerator designs

Whatever the combinations of the hardware building blocks, as long as they fit on the FPGA.

The FPGA resource consumption constraint:

$$\sum_i C_r(PE_i) + \sum_i C_r(FIFO_i) + C_r(infra) \leq Constraint_r,$$
$$\forall r \in \{BRAM, URAM, LUT, FF, DSP\}$$

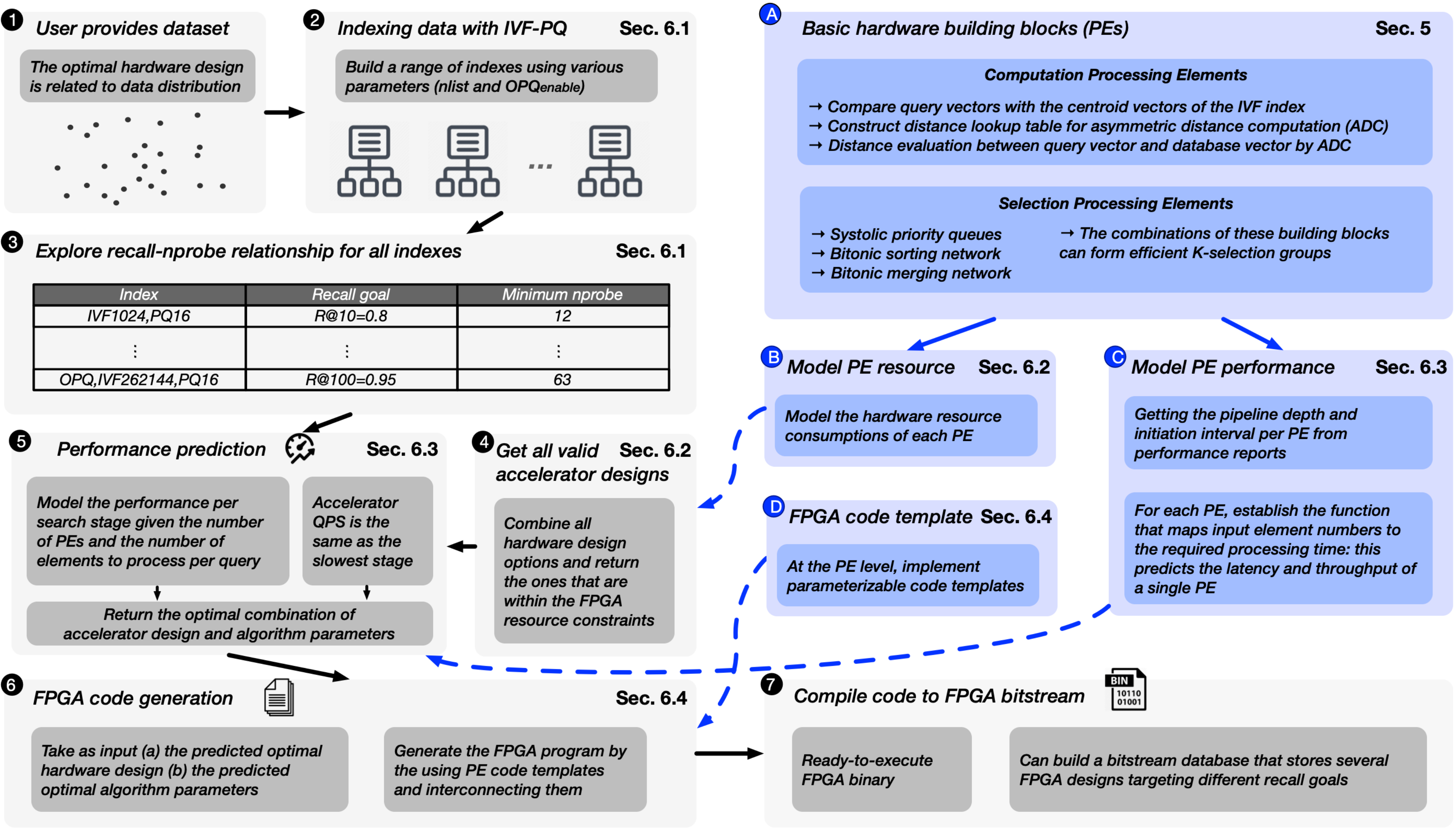
Performance prediction

Performance of a single processing element:

$$QPS_{PE} = freq/(L + (N-1) * II)$$

Performance of the entire accelerator depends on the slowest search stage:

$$QPS_{accelerator} = min(QPS_s), \text{where } s \in \{\text{Stages}\}$$



1 User provides dataset

The optimal hardware design is related to data distribution

2 Indexing data with IVF-PQ — Sec. 6.1

Build a range of indexes using various parameters (nlist and OPQ$_{enable}$)

3 Explore recall-nprobe relationship for all indexes — Sec. 6.1

Index	Recall goal	Minimum nprobe
IVF1024,PQ16	R@10=0.8	12
⋮	⋮	⋮
OPQ,IVF262144,PQ16	R@100=0.95	63

4 Get all valid accelerator designs — Sec. 6.2

Combine all hardware design options and return the ones that are within the FPGA resource constraints

5 Performance prediction — Sec. 6.3

Model the performance per search stage given the number of PEs and the number of elements to process per query

Accelerator QPS is the same as the slowest stage

Return the optimal combination of accelerator design and algorithm parameters

6 FPGA code generation — Sec. 6.4

Take as input (a) the predicted optimal hardware design (b) the predicted optimal algorithm parameters

Generate the FPGA program by the using PE code templates and interconnecting them

7 Compile code to FPGA bitstream

Ready-to-execute FPGA binary

Can build a bitstream database that stores several FPGA designs targeting different recall goals

A Basic hardware building blocks (PEs) — Sec. 5

Computation Processing Elements

→ Compare query vectors with the centroid vectors of the IVF index
→ Construct distance lookup table for asymmetric distance computation (ADC)
→ Distance evaluation between query vector and database vector by ADC

Selection Processing Elements

→ Systolic priority queues
→ Bitonic sorting network
→ Bitonic merging network

→ The combinations of these building blocks can form efficient K-selection groups

B Model PE resource — Sec. 6.2

Model the hardware resource consumptions of each PE

C Model PE performance — Sec. 6.3

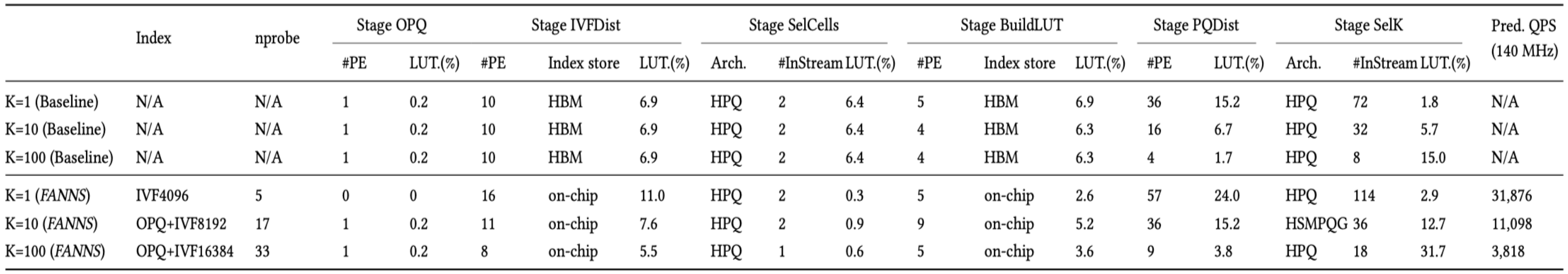
Getting the pipeline depth and initiation interval per PE from performance reports

For each PE, establish the function that maps input element numbers to the required processing time: this predicts the latency and throughput of a single PE

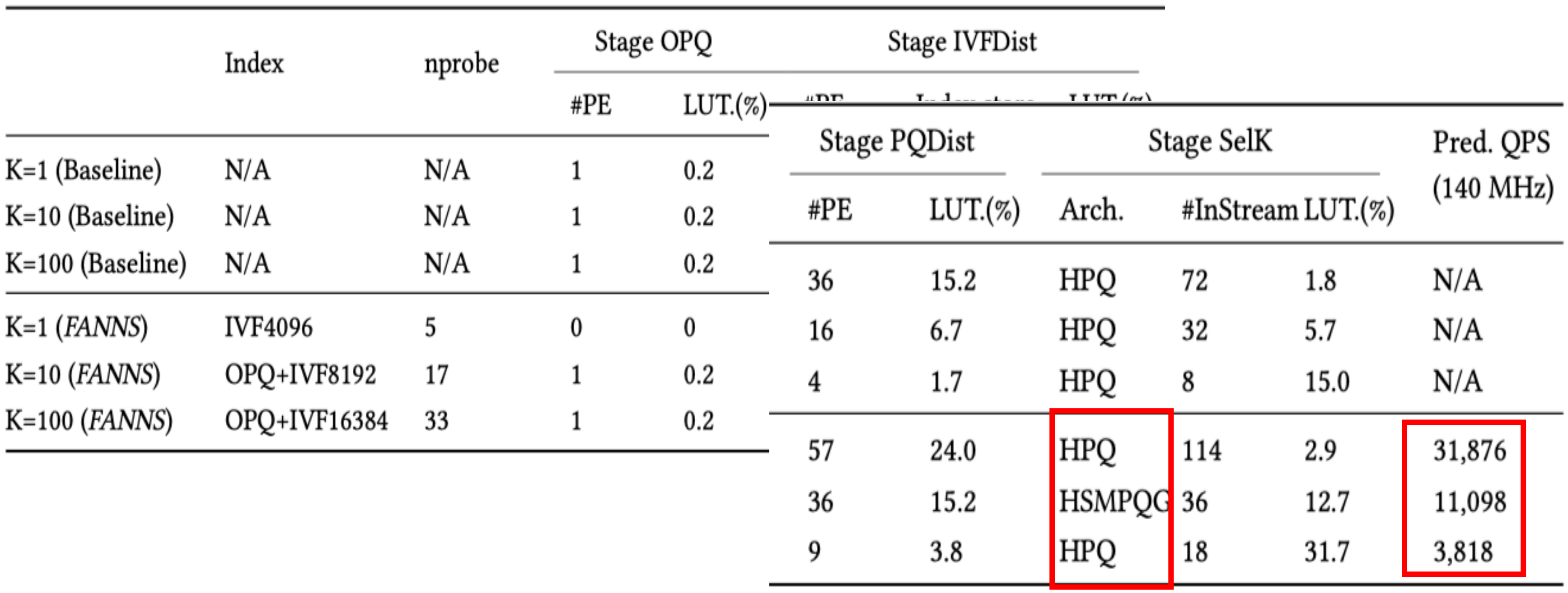
D FPGA code template — Sec. 6.4

At the PE level, implement parameterizable code templates

Some example designs



	Index	nprobe	Stage OPQ		Stage IVFDist			Stage SelCells			Stage BuildLUT			Stage PQDist		Stage SelK			Pred. QPS (140 MHz)
			#PE	LUT.(%)	#PE	Index store	LUT.(%)	Arch.	#InStream	LUT.(%)	#PE	Index store	LUT.(%)	#PE	LUT.(%)	Arch.	#InStream	LUT.(%)	
K=1 (Baseline)	N/A	N/A	1	0.2	10	HBM	6.9	HPQ	2	6.4	5	HBM	6.9	36	15.2	HPQ	72	1.8	N/A
K=10 (Baseline)	N/A	N/A	1	0.2	10	HBM	6.9	HPQ	2	6.4	4	HBM	6.3	16	6.7	HPQ	32	5.7	N/A
K=100 (Baseline)	N/A	N/A	1	0.2	10	HBM	6.9	HPQ	2	6.4	4	HBM	6.3	4	1.7	HPQ	8	15.0	N/A
K=1 (*FANNS*)	IVF4096	5	0	0	16	on-chip	11.0	HPQ	2	0.3	5	on-chip	2.6	57	24.0	HPQ	114	2.9	31,876
K=10 (*FANNS*)	OPQ+IVF8192	17	1	0.2	11	on-chip	7.6	HPQ	2	0.9	9	on-chip	5.2	36	15.2	HSMPQG	36	12.7	11,098
K=100 (*FANNS*)	OPQ+IVF16384	33	1	0.2	8	on-chip	5.5	HPQ	1	0.6	5	on-chip	3.6	9	3.8	HPQ	18	31.7	3,818



	Index	nprobe	Stage OPQ		Stage IVFDist
			#PE	LUT.(%)	
K=1 (Baseline)	N/A	N/A	1	0.2	
K=10 (Baseline)	N/A	N/A	1	0.2	
K=100 (Baseline)	N/A	N/A	1	0.2	
K=1 (*FANNS*)	IVF4096	5	0	0	
K=10 (*FANNS*)	OPQ+IVF8192	17	1	0.2	
K=100 (*FANNS*)	OPQ+IVF16384	33	1	0.2	

Stage PQDist		Stage SelK			Pred. QPS (140 MHz)
#PE	LUT.(%)	Arch.	#InStream	LUT.(%)	
36	15.2	HPQ	72	1.8	N/A
16	6.7	HPQ	32	5.7	N/A
4	1.7	HPQ	8	15.0	N/A
57	24.0	HPQ	114	2.9	31,876
36	15.2	HSMPQG	36	12.7	11,098
9	3.8	HPQ	18	31.7	3,818

Evaluation

Hardware

- CPU: Intel(R) Xeon(R) CPU E5-2686 v4 @ 2.30GHz, 16vCPU, 64GB
- FPGA: AMD Alveo U55c FPGA, 16 GB

Software

- Faiss: the most popular library for PQ-based ANN search
- Vitis HLS: for FPGA accelerator development

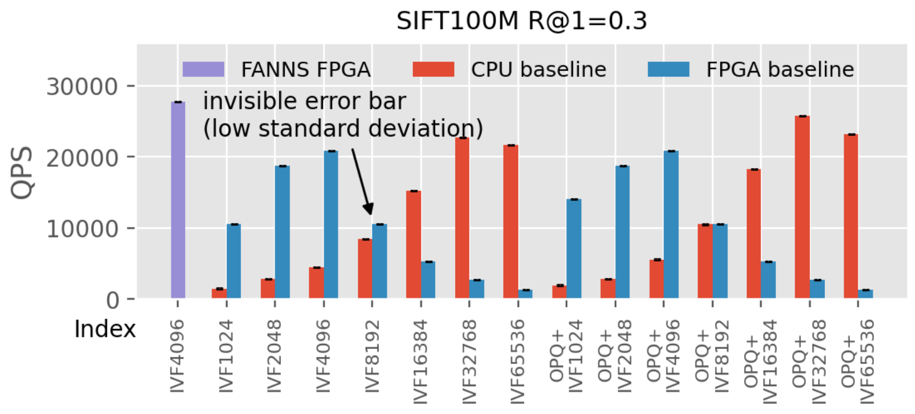
Dataset

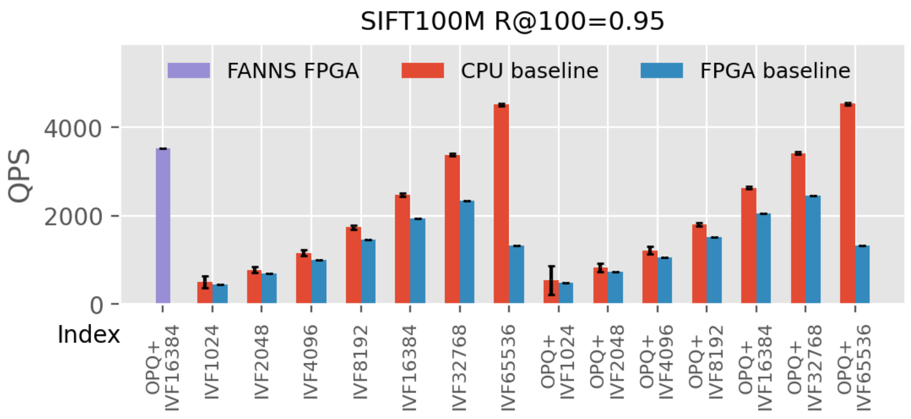
- SIFT: 128-dimensional, 100 million vectors
- Deep: 96-dimensional, 100 million vectors

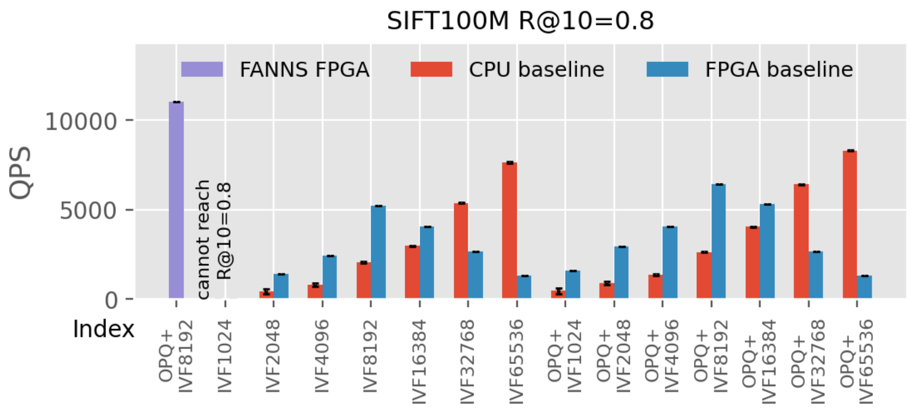
Throughput speedup over CPU and FPGA baselines

Up to 20.79x QPS as the FPGA baseline

Up to 29.98x QPS as the CPU baseline







Conclusion

Vector-based information retrieval is the future

The bottlenecks in the the IVF-PQ algorithm shift

FANNS: co-design hardware and algorithm

- Given a recall target on a dataset
- Use a performance-model to guide accelerator design
- Use a code-generator to make the design transparent to users