NEURAL INFORMATION PROCESSING SYSTEMS

Billion-scale Similarity Search Track 2: SSD Solution

Xiaomeng Yi, Xiaofan Luan, Weizhi Xu, Qianya Cheng, Jigao Luo, Xiangyu Wang, Jiquan Long, Xiao Yan, Zheng Bian, Jiarui Luo, Shengjun Li, Chengming Li

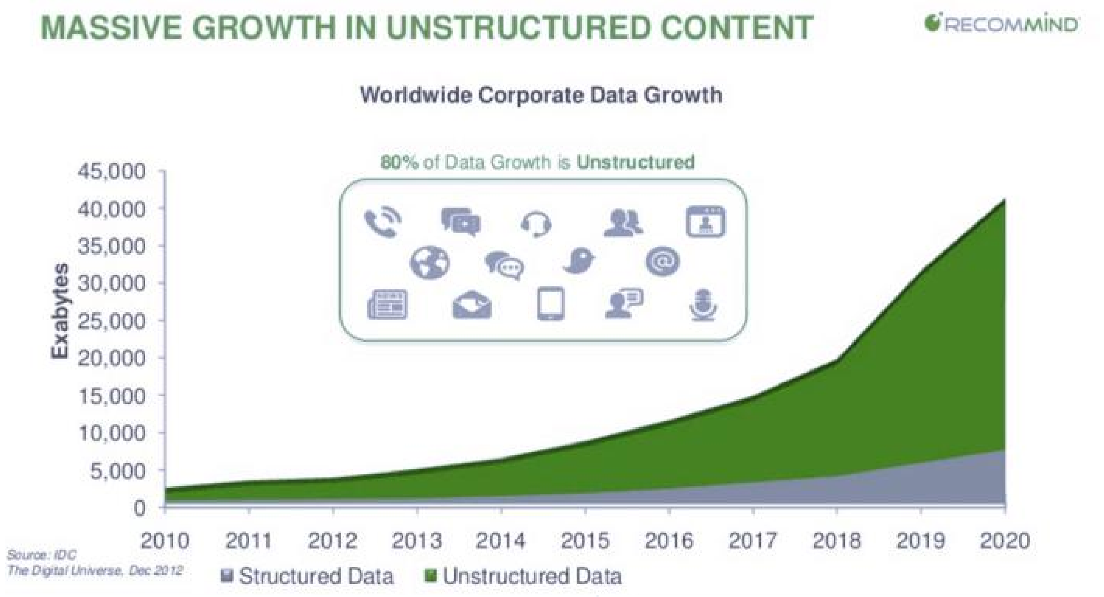
ZILLIZ

SOUTHERN UNIVERSITY OF SCIENCE AND TECHNOLOGY 南方科技大学

12/8/2021

Competition Background

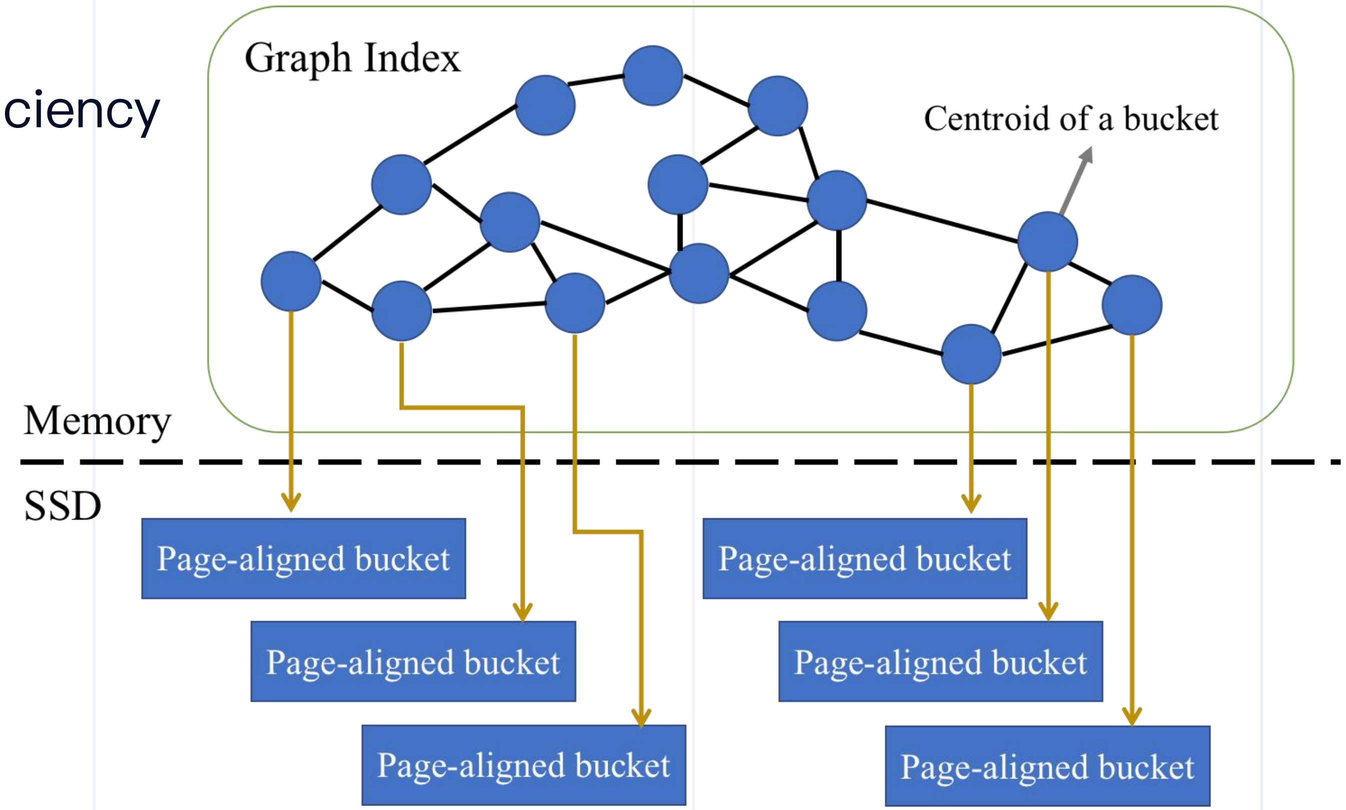
- **Motivation**
  - Fast data growth
  - Lower data management cost
  - Moderate performance degradation



- **Competition Requirements**
  - Billion-scale vector datasets (93GB~745GB)
  - Search: 1500+QPS
  - Index building: 4 days per dataset
  - Search Machine: 8 vCPU, 64 GB RAM, 1 TB SSD
  - Index Machine: 64 vCPU, 128 GB RAM, 4 TB SSD

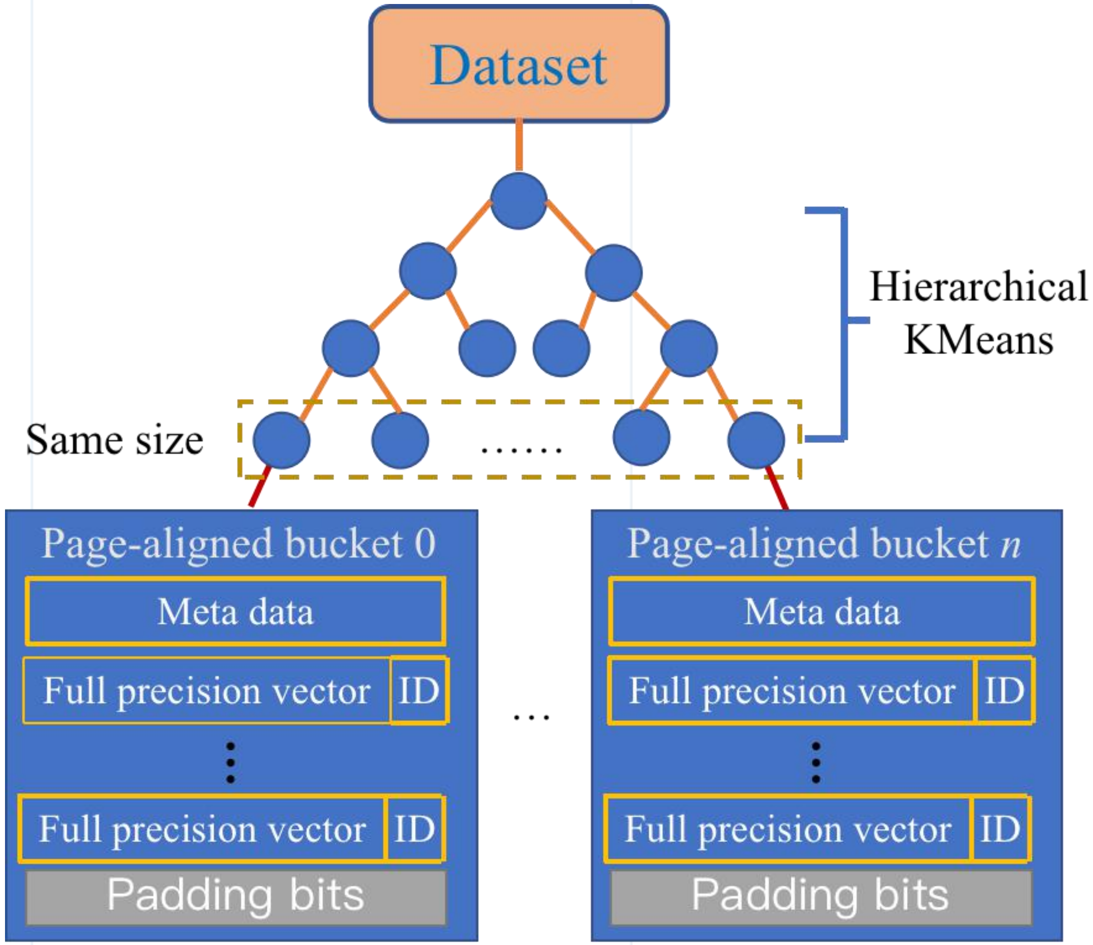
Solution Overview

- **Data storage in SSD**
  - Assign vectors to buckets
  - Buckets are page–aligned for read efficiency
- **Maintain bucket graph in memroy**
  - Represent each bucket with its centroid
  - Organize centroids in a graph index
- **Vector search**
  - Find related buckets through graph search
  - Fetch these buckets from SSD for scan



Assign vectors to buckets

- **Target**
  - Assign similar vectors into the same bucket
- **Key consideration**
  - Efficient read for SSD
  - Fast processing
- **Our choice:**
  - Hierachical KMeans
  - Page-aligned bucket size (4KB-8KB)

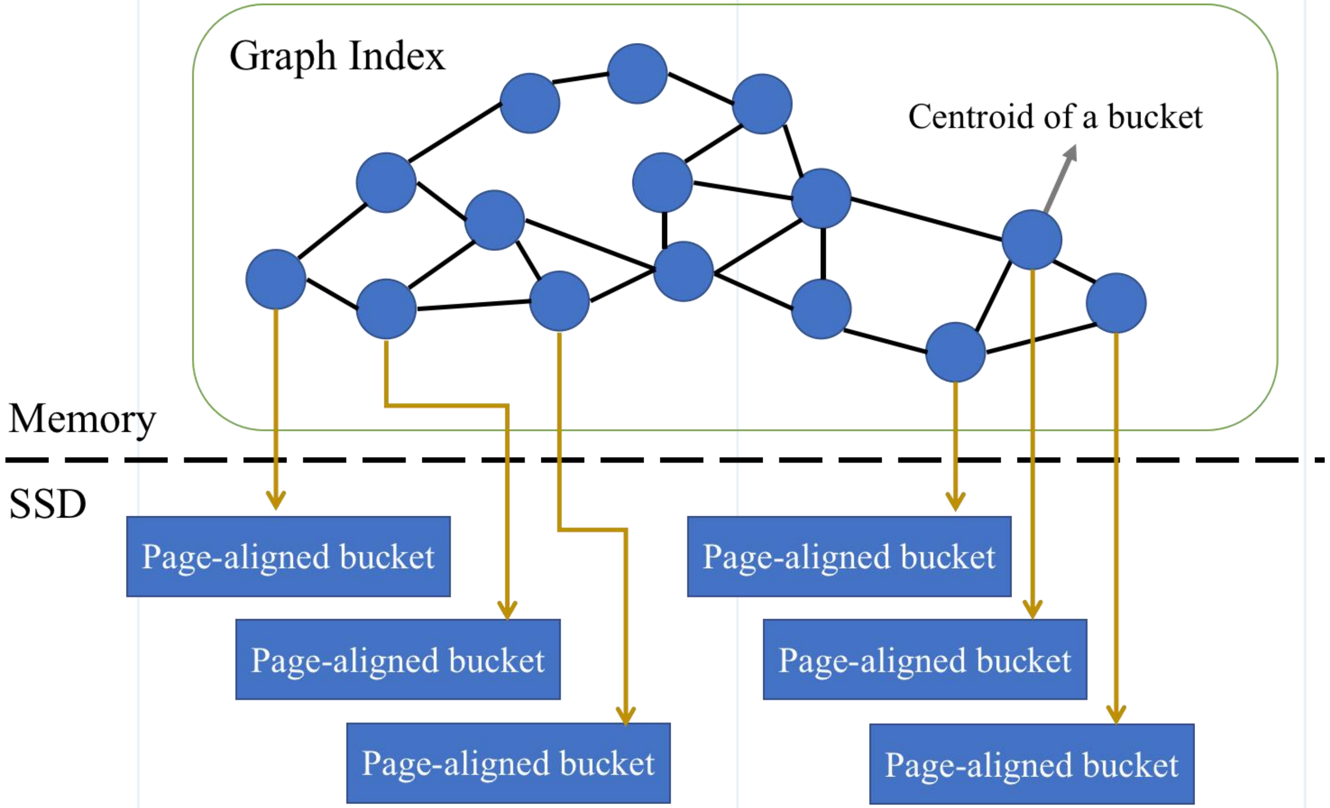


Index building

- **Target**
  - Quickly find query–related buckets

- **Key consideration**
  - Accuracy
  - Efficiency
  - Fit into memory

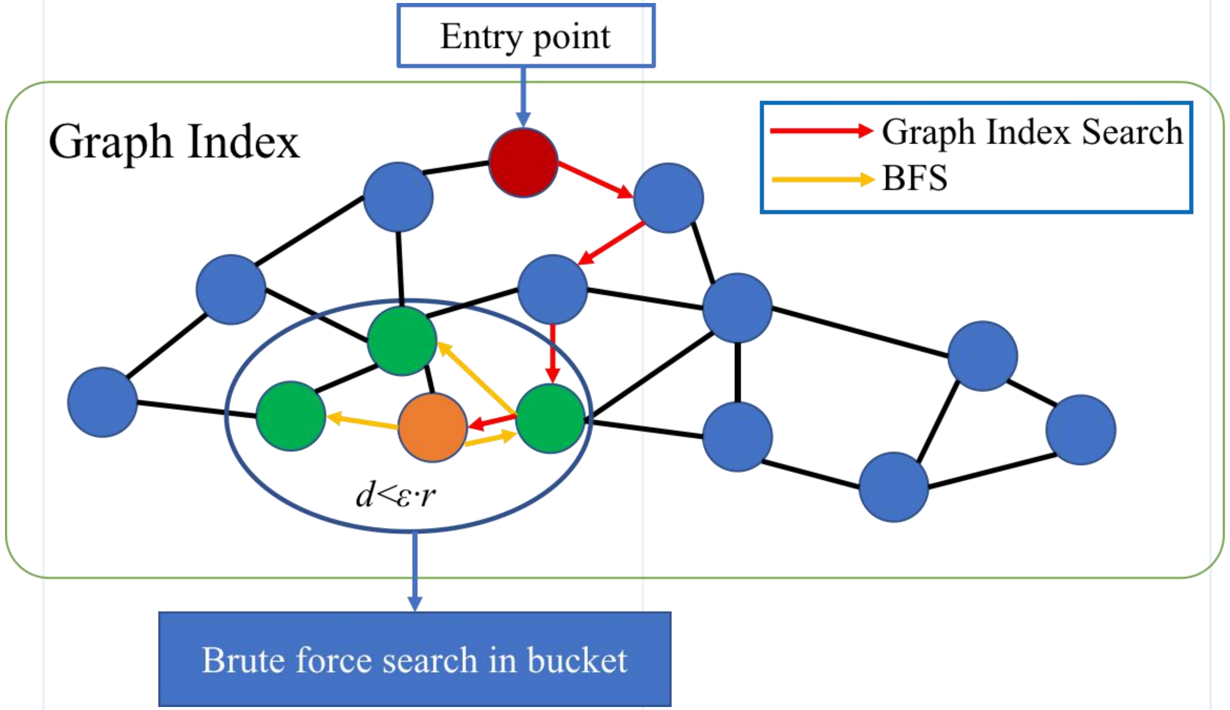
- **Our choice**
  - Graph index
  - Map centroid to integer vectors



Vector Search

- **Range Search**
  1. Search graph index to find a small set of seed buckets
  2. Find other related buckets from seed buckets with BFS until reaching range radius
  3. Brute-force scan related buckets

- **KNN Search**
  1. Search graph index to find related buckets
  2. Brute-force scan related buckets



- Optimization choices
  - Pruning
  - Data reuse across queries
  - Pipeline compute and I/O

Conclusion

- **Summary**
  - Inverted files with a graph index
  - Hierachical KMeans to speedup training
  - Page–aligned files to improve disk read efficiency
- **Results**
  - **Works better on range search dataset**
  - **Representativeness of centroids for range/KNN search queries**

Dataset	ssnpp-1B	text2image-1B	msspacev-1B
Recall@1500QPS(ours)	0.885(0.723↑)	0.495(0.007↑)	0.760(0.14↓)
Index building time(ours)	12 hrs	28 hrs	7 hrs
Recall@1500QPS(baseline)	0.162	0.488	0.901

Future directions

- Uniformed index to handle both range search and KNN search
- Analyze and exploit data hotness in queries
- Vector search with heterogeneous device/storage(NVM/SSD/GPU/FPGA)
- Distributed search algorithms

- Better understand datasets and indexes
- Automated index type/parameter recommendation
- Learned index for vectors

- Efficient vector search with attribute filtering
- Multi-modal information retrieval

About Zilliz

Vision

Build a data infrastructure that could help people accelerate AI adoptions in their organizations

Open-source Projects

- Milvus

Cloud native vector database for unstructured data

https://milvus.io

https://github.com/milvus-io/milvus

@milvusio

- Towhee

X2Vec: encode unstructured data into embeddings

https://hub.towhee.io/

https://github.com/towhee-io/towhee

@towheeio

Thank You

xiaomeng.yi@zilliz.com

https://www.linkedin.com/in/xiaomeng-yi-10b998a5/

@xiaomeng_y1