CHANNI: A Multi-Level Vector Search Index with Nested Graph Navigation

A Disk-Based ANN Index For Massively Scaled Apps

Nithin Mani (Cosdata)

2024-11-02

Contents

1 Abstract

CHANNI (Clustered Hierarchical Approximate Nested Navigable Index) introduces a novel vector indexing architecture that bridges the gap between memory-efficient clustering and high-performance graph navigation. By combining a hierarchical navigable small world graph at the cluster level with hierarchical navigable graphs within clusters, CHANNI achieves superior performance while maintaining efficient resource utilization for large-scale vector search applications.

2 Introduction

Vector similarity search at scale presents significant challenges in balancing memory usage, search speed, and result quality. Traditional approaches either consume excessive memory like pure HNSW implementations, or compromise on accuracy through quantization-based methods. CHANNI addresses these limitations through a novel multi-level architecture that intelligently combines clustering with navigable graphs at different granularities.

3 Core Innovation

3.1 Multi-Level Navigation Architecture

The fundamental innovation in CHANNI lies in its dual-level navigation approach. At the top level, a hierarchical navigable small world graph connects cluster primaries, enabling rapid identification of relevant clusters. Within each cluster, a hierarchical navigable graph structure facilitates efficient local search with high precision. This separation allows for independent parameter tuning at each level, optimizing both coarse and fine-grained search operations while maintaining an optimal balance between disk usage and memory consumption.

3.2 Primary-Based Cluster Representation

A key innovation in CHANNI is using actual vectors from the dataset as cluster representatives (primaries), rather than maintaining computed centroids. Unlike traditional clustering approaches like k-means that continuously calculate and update centroids, CHANNI uses these primary vectors for both top-level HNSW navigation and intra-cluster organization.

3.2.1 Advantages

This design choice offers several significant benefits:

- Eliminates continuous computational overhead of centroid calculations during normal operations
- Avoids centroid drift during routine updates
- Ensures cluster representative is always a real vector from the dataset
- Reduces memory footprint by not storing additional centroid vectors
- Maintains stable cluster identification through primary vectors

3.2.2 Hybrid Approach for Splits

CHANNI employs a strategic hybrid approach during cluster splits:

- Computes temporary centroids only during split operations
- Selects new primaries as vectors nearest to computed centroids
- One-time computation cost justified by improved cluster quality
- Better balance in resulting sub-clusters
- More representative space partitioning

3.2.3 Trade-offs

The system balances computational efficiency with cluster quality:

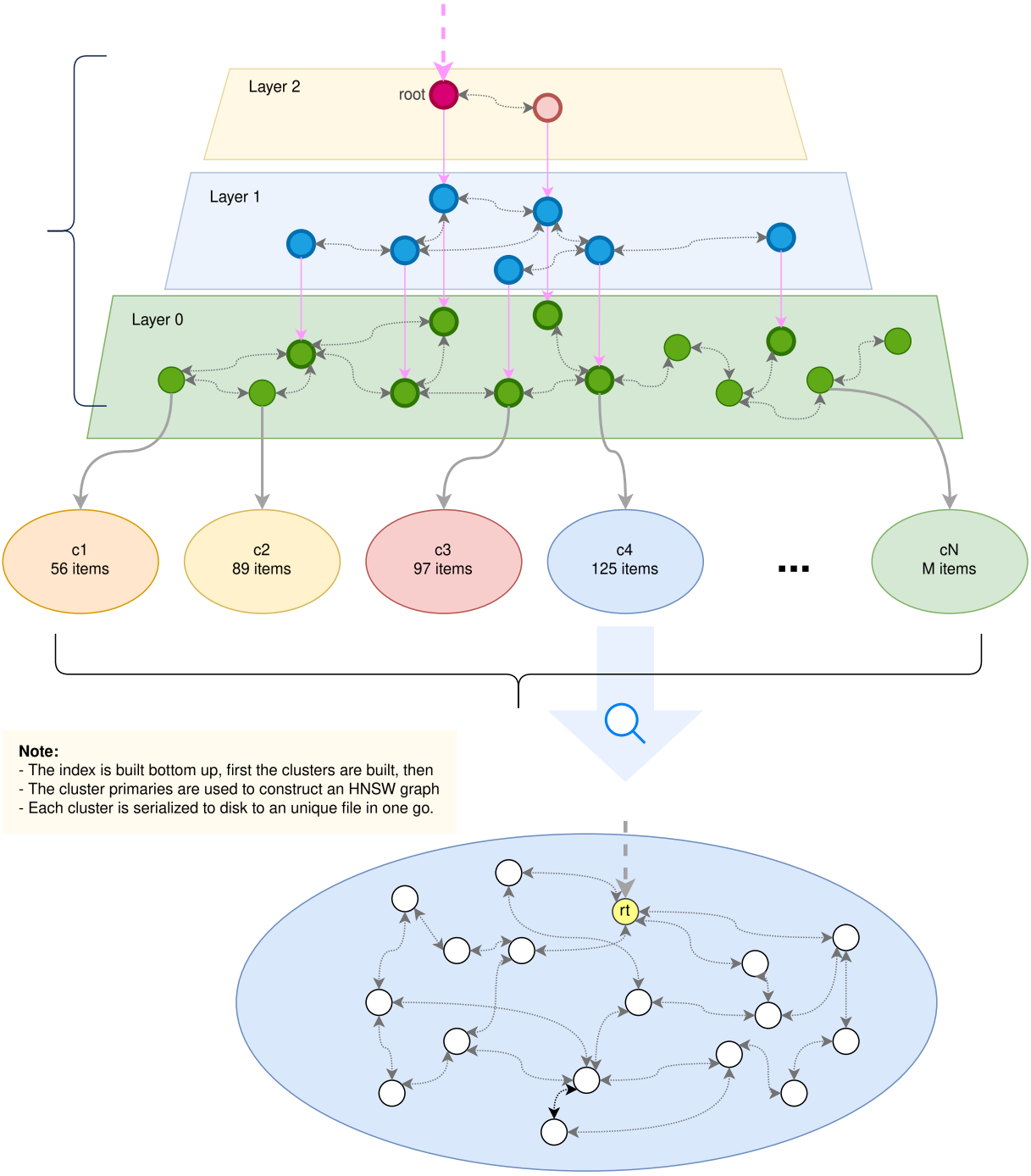
- Normal operations maintain zero centroid computation overhead
- Split operations accept one-time computation cost for better clustering
- Quality of initial cluster formation depends on primary selection strategy
- Split quality improved through selective centroid computation
- Memory impact limited to temporary split calculations

3.2.4 Implementation Impact

This representation strategy affects several system components:

- Top-level HNSW graph connects actual data vectors (primaries)
- Cluster splits use temporary centroid computation for better primary selection
- Update operations maintain stable primaries unless splits/merges occur
- Split operations can be scheduled during system low-load periods

4 Illustration



5 Clustering Strategy: Vector Space Partitioning through Representative Sampling

5.1 Initial Cluster Primary Selection

CHANNI implements a distinctive approach to cluster formation that challenges traditional clustering methods. Instead of employing computationally expensive k-means clustering, the system defers index construction until a substantial portion of the vector embeddings are available. It then performs large-scale sampling to select cluster primaries, which serve as initial primaries. This strategy leverages the natural distribution properties of the vector space, achieving well-distributed cluster sizes without the overhead of iterative centroid refinement.

5.2 Theoretical Analysis

The distribution of cluster sizes in CHANNI warrants careful examination. When selecting initial cluster primaries from a large sample of the vector space, the resulting cluster size distribution is influenced by:

- The inherent distribution of the vector embeddings in the space
- Local density variations in different regions of the vector space
- The dimensionality and characteristics of the embedding space

Since vectors are assigned to their closest cluster primary without a similarity threshold, the cluster sizes naturally emerge from the interplay between:

- The initial sampling distribution of primaries
- The density distribution of vectors in different regions
- The relative distances between cluster primaries

The absence of a similarity threshold means cluster boundaries are purely determined by relative proximity to primaries. This raises interesting questions about:

- How the initial sampling density of primaries affects final cluster size distribution
- Whether certain regions of the vector space tend to accumulate larger clusters
- The relationship between vector space topology and resulting cluster characteristics
- Potential strategies for sampling primaries to achieve more balanced cluster sizes

An important research direction is to empirically analyze:

- Typical cluster size distributions across different types of vector spaces

- The relationship between sampling ratio and cluster size variation
- The impact of non-uniform cluster sizes on search performance
- Whether adaptive sampling strategies could improve size distribution without sacrificing the computational efficiency of our approach

6 Index Construction and Performance

6.1 Build Process Overview

The construction of CHANNI index follows a distinct phased approach:

6.1.1 Initial Sampling Phase

- Waiting for substantial vector collection (80% of expected data)
- Large-scale sampling to select cluster primaries
- Time complexity: $O(n)$ for sampling from n vectors

6.1.2 Cluster Formation Phase

- Assignment of vectors to nearest primaries via HNSW routing
- Progressive cluster building as vectors are assigned
- Parallel processing capabilities during assignment
- Time complexity: $O(n \log k)$ where k is number of clusters

6.1.3 Graph Construction Phase

- Building top-level HNSW graph connecting cluster primaries
- Construction of hierarchical navigable graphs within each cluster
- Independent parameter tuning for each level
- Time complexity: $O(k \log k)$ for cluster primaries' HNSW, $O(m \log m)$ for each cluster of size m

6.2 Build-Time Optimizations

- Parallel cluster construction
- Efficient memory management during building
- Batch processing of vector assignments
- Progressive disk serialization of completed clusters

6.3 Relationship to Search Performance

The index construction process directly influences search efficiency:

- Quality of initial primary selection affects cluster balance
- Cluster primaries' HNSW structure determines routing efficiency
- Within-cluster graph connectivity impacts local search speed
- Build-time parameter choices influence search-time performance trade-offs

6.4 Resource Requirements

- Memory usage patterns during construction
- Disk I/O characteristics for cluster serialization
- CPU utilization during different phases
- Temporary storage needs

7 Technical Architecture

7.1 Search Process

The search process in CHANNI occurs in multiple phases, each optimized for its specific role. Initially, the system navigates the HNSW graph to identify relevant clusters. Once target clusters are identified, the system accesses their hierarchical navigable graphs to perform precise local searches. This multi-phase approach ensures both broad coverage and detailed local exploration while maintaining efficient resource utilization.

7.2 Parameter Configuration

CHANNI's architecture enables fine-grained control over search and construction parameters at each level. The top-level HNSW graph can be optimized for quick cluster identification, while cluster-level graphs can be tuned for precision. This flexibility allows for sophisticated trade-off management between search speed and accuracy.

8 Dynamic Index Maintenance

8.1 Split & Merge Operations

8.1.1 Cluster Split Strategy

When clusters grow beyond predetermined size thresholds, CHANNI employs an adaptive split mechanism with OOD detection:

1. Initial Centroid Formation
    - Compute initial split count $N = \max(3, \log(\text{cluster\_size}))$
    - Use mini-batch k-means to form N temporary centroids
    - This allows detection of potential subclusters and OOD vectors
2. OOD Detection & Handling
    - Calculate mean distance μ and standard deviation σ of vectors to their nearest centroid
    - Flag vectors with distance $> \mu + 2\sigma$ as potential OOD
    - Create separate OOD clusters if sufficient OOD vectors are detected
    - Limit OOD cluster size to prevent fragmentation
3. Final Split Process
    - For non-OOD vectors, merge centroids until reaching target split count (usually 2-3)
    - Select actual vectors nearest to final centroids as new primaries
    - Assign remaining vectors to nearest primary
4. Primary Selection & Integration
    - Choose vectors nearest to computed centroids as new primaries
5. Efficient HNSW routing graph updates
    - CHANNI employs an innovative strategy to optimize cluster splits while maintaining routing efficiency. During a split operation, the system reuses the original cluster's primary-hash based file name for one of the new centroids, specifically selecting the centroid closest to the original primary. This approach requires only updating the cluster's internal root to point to the new centroid vector, while creating a new cluster file only for the additional centroid.
    - The selection of the nearest centroid for reuse is crucial for maintaining the effectiveness of the HNSW routing graph. By preserving locality in the graph traversal, vectors previously routable via the original primary remain accessible through similar paths. This strategy ensures that distance-based routing decisions remain valid and minimizes disruption to the existing HNSW graph structure while preserving locality-sensitive navigation patterns.
    - This optimization dramatically reduces maintenance overhead by avoiding expensive HNSW deletion operations and maintaining unidirectional neighbor relationships. The approach requires just a single update operation instead of costly deletion and multiple insertions, while preserving the existing cluster file structure.

8.1.2 Cluster Merge Operations

Conversely, CHANNI identifies and merges undersized clusters, which may result from:

- Out-of-distribution vectors forming small clusters
- Deletion operations leaving sparse clusters

- Natural data distribution shifts

The merge process involves:

- Size threshold monitoring during transaction commits
- Identification of merge candidates based on size and proximity
- Cluster consolidation with primary selection from merged set

1. Merge Operation Strategy
    - For cluster merges, CHANNI implements a dual-primary routing approach that maintains multiple valid paths to merged clusters. Instead of deleting redundant primaries, the system updates primary-hash to cluster mappings for both original primaries to reference a single consolidated cluster file. The cluster's root is updated to reflect the new primary while maintaining both original routing paths in the HNSW graph.
    - This dual-primary strategy is particularly crucial for handling out-of-distribution (OOD) vectors. Since OOD clusters are typically smaller and more likely to undergo merges, maintaining multiple routing paths ensures reliable vector retrieval. OOD vectors, which may be reachable through different routing paths in the HNSW graph, remain accessible through either primary. This approach prevents potential accessibility issues during cluster consolidation while supporting efficient retrieval of outlier vectors.
    - The preservation of multiple valid entry points to merged clusters ensures stability in the HNSW routing structure and maintains search quality across diverse vector distributions. This strategy eliminates the need for HNSW graph deletions, simplifies recovery operations, and significantly reduces maintenance overhead while ensuring robust accessibility for all vectors, including outliers and OOD cases.

8.1.3 Transaction Management

Both split and merge operations are integrated into the transaction commit process:

- Atomic updates ensure index consistency
- Batch processing of multiple operations when possible
- Efficient handling of concurrent operations
- Recovery mechanisms for interrupted operations

8.1.4 Performance Implications

The dynamic maintenance strategy impacts several aspects:

- Search performance during reorganization
- Resource utilization during splits and merges

- HNSW graph quality maintenance
- Overall index balance and efficiency

This adaptive approach ensures CHANNI maintains optimal performance characteristics even as the underlying data distribution evolves over time.

8.2 Parallel Maintenance Operations

8.2.1 Concurrent Cluster Operations

CHANNI's architecture enables parallel execution of both split and merge operations, maximizing throughput during index maintenance. Multiple large clusters can undergo split operations simultaneously, while independent merge operations can process smaller clusters in parallel. This concurrent processing capability significantly reduces maintenance windows and keeps the index optimized even under heavy update loads.

The system employs fine-grained locking mechanisms to ensure data consistency during parallel operations. When splitting large clusters, each operation works independently on its target cluster while maintaining HNSW graph consistency. Similarly, merge operations can process multiple sets of small clusters concurrently, with careful coordination to prevent conflicts in primaries management.

Key aspects of parallel processing include:

- Independent primary sampling and vector reassignment for splits
- Simultaneous merge operations for non-overlapping cluster sets
- Parallel updates to the HNSW routing structure
- Transaction isolation between different maintenance operations

8.2.2 Resource Coordination

The parallel maintenance subsystem carefully manages system resources to balance maintenance tasks with ongoing search operations. A sophisticated resource manager allocates processing power, memory, and I/O bandwidth across concurrent operations while ensuring search performance remains within acceptable bounds.

The system employs multi-version concurrency control (MVCC) to maintain consistency during parallel operations. This approach allows read operations to proceed unimpeded while maintenance tasks modify cluster structures. Transaction boundaries are carefully managed to ensure atomic updates and provide clear rollback points if needed.

Critical resource considerations:

- Worker pool allocation between search and maintenance tasks
- Memory management for concurrent operations
- I/O scheduling and cache coherency
- CPU quota distribution across parallel tasks

8.2.3 Monitoring and Recovery

A comprehensive monitoring system tracks the progress and health of parallel maintenance operations. Real-time monitoring enables the system to detect and respond to resource constraints or failed operations quickly. If a maintenance operation fails, the system can roll back changes while keeping other parallel operations unaffected.

The parallel maintenance capability significantly improves CHANNI's ability to handle dynamic workloads, ensuring the index remains optimized without impacting search availability. By carefully balancing resources and maintaining consistency, the system provides robust performance even during intensive maintenance periods.

9 System Implementation

9.1 Resource Management

The implementation focuses on efficient resource utilization through careful memory management and disk I/O optimization. Cluster data resides primarily on disk, with active clusters cached in memory as needed. The system maintains two specialized registries: a lightweight node registry for HNSW navigation and a cluster registry for data access, both optimized for their specific access patterns.

10 Performance Characteristics

10.1 Search Efficiency

The multi-level structure of CHANNI enables highly efficient search operations. The top-level HNSW graph of cluster primaries provides rapid cluster identification in logarithmic time complexity relative to the number of clusters. Within each cluster, the hierarchical navigable graph structure enables precise local search, maintaining high recall while minimizing distance computations.

Each level can be independently tuned through its efSearch parameter:

- Cluster-level navigation: Controls the trade-off between search speed and cluster selection accuracy
- Within-cluster search: Manages the precision of local vector retrieval

10.2 I/O Patterns

CHANNI's architecture is specifically designed for efficient disk I/O patterns:

- Sequential cluster reads minimize disk seek operations
- Single-file cluster serialization enables efficient disk access
- Smart caching strategies keep frequently accessed clusters in memory
- Asynchronous I/O capabilities for parallel cluster loading

11 Search Parallelization Strategy

11.1 Distributed Search Architecture

CHANNI's cluster-based design inherently supports efficient parallel search operations through a map-reduce style architecture. The search process begins with a mapping phase where the query is routed through the cluster primaries' HNSW graph to identify relevant clusters. This initial routing process, while sequential, quickly narrows down the search space to a subset of promising clusters.

Once target clusters are identified, CHANNI distributes search operations across multiple workers for parallel execution. Each worker independently processes the search within its assigned clusters, leveraging the local hierarchical navigable graphs for precise vector retrieval. This parallel execution phase significantly reduces overall search latency, especially for high-recall scenarios requiring exploration of multiple clusters.

The reduce phase aggregates results through a sophisticated merging process. A priority queue-based mechanism combines candidates from different clusters, ensuring the global top-k results are accurately identified. This phase includes distance-based filtering and progressive refinement to maintain search quality while minimizing resource usage.

Key performance optimizations include:

- Dynamic worker assignment based on cluster sizes and system load
- Cluster affinity mechanisms for optimal cache utilization
- Adaptive batch processing for query efficiency

11.2 Resource Management

The parallel search architecture requires careful resource coordination across workers. CHANNI implements a memory-aware scheduling system that optimizes cluster cache utilization across parallel searches. This includes:

- Coordinated cache management across search workers
- Intelligent prefetching based on query patterns
- Efficient result buffer allocation and management

The system dynamically adjusts its parallelization strategy based on:

- Available computational resources
- Current system load
- Query characteristics
- Desired recall targets

11.3 Scaling Behavior

CHANNI's parallel search capability demonstrates near-linear scaling with additional search workers, particularly for large-scale deployments. The cluster-based architecture minimizes cross-worker communication overhead, allowing efficient resource utilization even with high parallelization factors.

The effectiveness of parallelization is influenced by several factors:

- Cluster size distribution
- Network topology and bandwidth
- Cache hit rates
- Query complexity

Through careful tuning of these parameters, CHANNI maintains high search performance while efficiently utilizing available computational resources across distributed environments.

12 Scalability Features

12.1 Horizontal Scaling

The cluster-based organization naturally supports distributed deployment:

- Clusters can be distributed across multiple machines
- Top-level HNSW graph serves as an efficient routing mechanism
- Independent cluster processing enables true parallelism
- Load balancing through intelligent cluster distribution

12.2 Resource Adaptation

The system adapts to various hardware configurations through:

- Configurable cache sizes for different memory profiles
- Adjustable cluster sizes based on available resources
- Tunable graph parameters for performance optimization
- Flexible disk-memory trade-off management

13 Experimental Results

13.1 Dataset Characteristics

[Details about benchmark datasets used for evaluation]

13.2 Performance Metrics

13.2.1 Search Quality

- Recall@k measurements across different k values
- Precision comparison with pure HNSW and IVF approaches
- Quality-speed trade-off analysis

13.2.2 Resource Utilization

- Memory consumption patterns
- Disk I/O measurements
- CPU utilization analysis
- Cache hit rates

13.2.3 Scalability Tests

- Performance scaling with dataset size
- Distribution efficiency measurements
- Parallel processing capabilities

14 Future Research Directions

14.1 Theoretical Developments

- Mathematical modeling of cluster size distributions
- Optimization of sampling strategies
- Analysis of vector space characteristics impact
- Performance bounds theoretical analysis

14.2 Technical Enhancements

- Dynamic parameter adjustment mechanisms
- Advanced caching strategies
- Distributed system optimizations
- Update handling improvements

14.3 Potential Applications

- Cross-modal vector search
- Dynamic vector collections
- Real-time search requirements
- Resource-constrained environments

15 Advantages over Existing Approaches

CHANNI demonstrates significant advantages over both pure HNSW and inverted file approaches. Compared to pure HNSW, it achieves dramatically reduced memory requirements while maintaining high search quality. Unlike quantization-based methods, CHANNI preserves vector precision within clusters and eliminates the need for expensive clustering computations through its innovative sampling-based approach.

16 Conclusion

CHANNI represents a significant advancement in large-scale vector search technology. Its innovative approach combining efficient sampling-based clustering with multi-level navigation provides a practical solution to the challenges of scale, performance, and resource utilization. The system's flexibility and scalability make it particularly valuable for real-world applications requiring high-performance vector similarity search.

17 Acknowledgments

[To be added]

18 References

[To be added based on relevant literature and technical foundations]

19 TODO

- Out of distribution vectors can be kept in exclusive clusters
- Versioning
- Metadata filtering