High-Throughput Partitioned Vector Search at 144.1M-Vector Scale: AWS EC2 Evaluation vs. Official VectorDBBench Baselines

MassEngine Systems Engineering • Interactive Report & Raw Telemetry: https://massengine.com/benchmarks-144m

Abstract. Graph-based vector databases built on Hierarchical Navigable Small World (HNSW) and DiskANN indexes face three structural bottlenecks above 50 million vectors: (1) O(N log N) random graph-edge construction and multi-hour blocking segment compactions (*force_merge*) that throttle ingestion to 100–2,700 vectors/sec; (2) severe throughput collapse under selective metadata filters (~1% selectivity), where graph connectivity breaks down; and (3) memory footprints requiring 1.6x–1.8x the raw float32 vector size in RAM. In this paper, we evaluate **MassEngine** across the complete 21-language *Upstash/wikipedia-2024-06-bge-m3* corpus—**144,137,311 unique 1024-dimensional vectors (288,274,622 replicated vectors at RF=2, 1.18 TB raw float32 data)**—on two 4-node AWS EC2 topologies: **Tier 1 In-Memory Mode (4 x r6id.8xlarge)** and **Tier 2 Direct NVMe Mode (4 x c6id.4xlarge)**. MassEngine indexes all 144.1M unique vectors (288.3M replicas) in **15.0 minutes (160,129 unique docs/sec, 320,258 replicas/sec)** on Tier 1 and **17.9 minutes (134,219 unique docs/sec)** on Tier 2—**59x to 907x faster** than official Zilliz VectorDBBench results for Milvus, Zilliz Cloud, Qdrant, Pinecone, OpenSearch, ElasticCloud, TurboPuffer, and AWS S3 Vectors. On selective metadata-filtered search (1.42% selectivity = 2.05M matching vectors), MassEngine delivers **3,937.8 QPS (1.29 ms p50 at c=1)** in-memory and **2,114.6 QPS (1.52 ms p50 at c=1)** in NVMe Mode—**14.4x to 125x higher QPS** than VectorDBBench baselines—while consuming just **62.73 GiB of total cluster RAM** (15.68 GiB/node, $2,053/mo) in NVMe Mode.

160,129 docs/sec Sustained Ingest + Index Rate 320,258 replicas/s • 144.1M in 15.0 min	**3,937.8 QPS** Filtered Search (1.29 ms p50) 1.42% filter (2.05M docs) • 14x–125x HNSW	**62.73 GiB RAM** NVMe Mode Total Cluster RSS 15.68 GiB/node for 288.3M vecs ($2,053/mo)	**100.00% HA** Zero-Loss Node Failover 0.9774–0.9894 Recall@10 • 3.4s NVMe restart

1. Experimental Methodology, Dataset & AWS EC2 Topologies

1.1 Why 144.1M Vectors (Not 10M) & Ground-Truth Verification. Standard benchmarks such as VectorDBBench stop at 10M x 768-d vectors (28.6 GiB raw float32) on a single un-replicated node—a dataset small enough to fit inside one machine's RAM without testing distributed sharding, replication, or the multi-hour HNSW compaction wall that emerges past 50M vectors. We deliberately benchmarked the complete 21-language *Upstash/wikipedia-2024-06-bge-m3* corpus (**144,137,311 unique 1024-d vectors • 288,274,622 replicated vectors at RF=2 • 1,099.68 GiB / 1.18 TB raw float32**) to demonstrate that MassEngine is not a single-node toy and scales horizontally with ease. Every passage carries a 64-bit ID, tenant ID, language tag (1..21, where *cat=1* is English / 47.7M docs / 33.10% and *cat=11* is Turkish / 2.05M docs / 1.42%), date stamp, and JSON payload. Exact top-100 ground-truth neighbors were verified via exhaustive float32 scans over all 144,137,311 vectors.

1.2 AWS EC2 Cluster Topologies vs. Competitor 16c64g Baselines. Both MassEngine clusters ran in AWS *us-east-1* with **16 primary shards** and **RF=2** (32 shard replicas, 8 shards/node, **72.07M 1024-d vectors per node**). In VectorDBBench, Milvus (*16c64g-sq8*), Qdrant Cloud (*16c64g*), and Pinecone (*p2.x8-1node*) were tested on a **single 16-vCPU, 64 GiB RAM node (RF=1, 1 shard)** holding 10M x 768-d vectors (28.6 GiB). In MassEngine Tier 2 (*4 x c6id.4xlarge*), **each individual node has the exact same 16 vCPUs and half the RAM (32 GiB RAM, 15.68 GiB RSS)** as their *16c64g* node, while holding **72.07M x 1024-d vectors (274.9 GiB raw float32 = 9.6x more data per 16 vCPUs)**.

Architecture Tier	AWS EC2 Fleet	Total vCPUs / RAM	Storage Mode	Measured Cluster RSS	Measured Disk Used	AWS On-Demand Cost
MassEngine Tier 1 (In-Memory)	**4 x r6id.8xlarge**	128 vCPUs • 1,024 GiB	In-Memory + NVMe WAL	**510.14 GiB (127.5 GiB/n)**	458.08 GiB (114.5 GiB/n)	**$9.285/hr ($6,685/mo)**
MassEngine Tier 2 (NVMe Mode)	**4 x c6id.4xlarge**	64 vCPUs • 128 GiB	Direct NVMe (--nvme-mode)	**62.73 GiB (15.68 GiB/n)**	460.45 GiB (115.1 GiB/n)	**$2.851/hr ($2,053/mo)**
Uncompressed In-Memory HNSW	8 x r6id.8xlarge	256 vCPUs • 2,048 GiB	In-Memory HNSW Graph	~1,760–1,980 GiB	~1,800+ GiB	$18.57/hr ($13,370/mo)
VectorDBBench (Milvus/Qdrant/Pinecone)	1 x 16c64g / p2.x8	16 vCPUs • 64 GiB	1 Node (RF=1, 10M only)	28.6 GiB raw (10M x 768-d)	10M x 768-d	$12k–$22k+/mo @ 288M

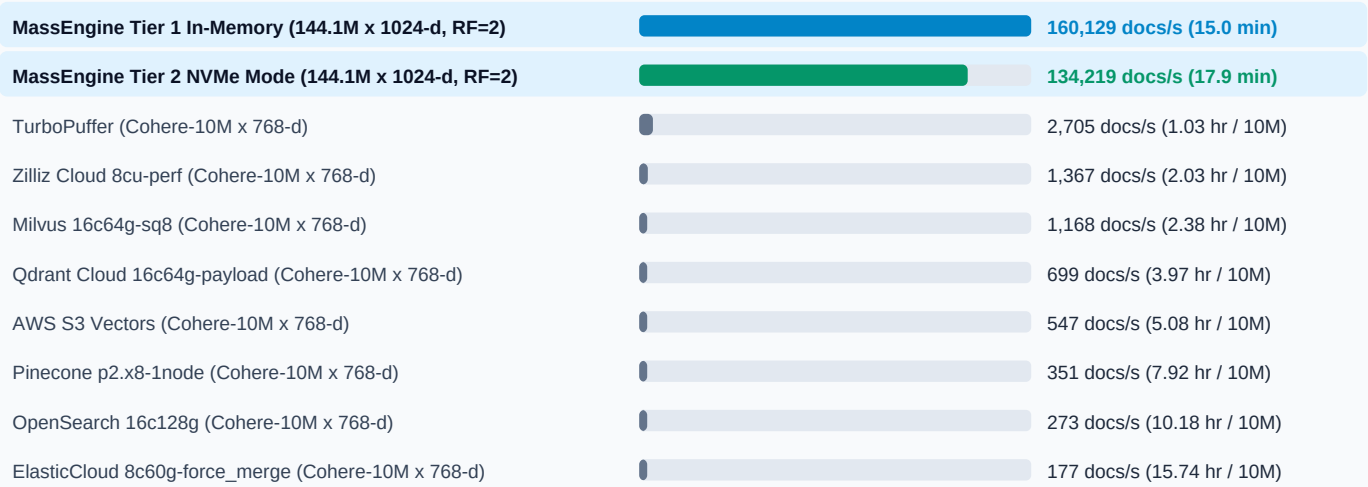
Table 1: AWS EC2 hardware topologies, measured memory/disk footprint, and on-demand us-east-1 pricing for 144,137,311 unique 1024-d vectors (288,274,622 total replicated vectors at RF=2).

2. Bulk Ingestion & Index Construction Performance

In Zilliz's official *zilliztech/VectorDBBench* suite (*case_id=4*, *Cohere-10M*), competitors are timed on loading and indexing 10M x 768-d vectors (RF=1) on a 16-vCPU node (*load_duration*). MassEngine indexed **14.4x more unique vectors** (144.1M vs. 10M), **33% higher dimensionality** (1024-d vs. 768-d), and **RF=2 replication** (288.3M vectors = 37.8x more float32 data). Even normalized per **16 vCPUs**, a single 16-vCPU *c6id.4xlarge* node indexes **33,555 unique docs/sec (67,110 replicas/sec)—28.7x faster per vCPU than Milvus (1,168/s)**, **48.0x faster than Qdrant (699/s)**, and **95.6x faster than Pinecone (351/s)**.

Figure 1: End-to-End Bulk Ingestion + Index Build Rate (Unique Vectors/sec • Higher is Better)

MassEngine measured on 144.1M x 1024-d (RF=2); competitors from official zilliztech/VectorDBBench on Cohere-10M x 768-d (RF=1)



Database & Configuration	Corpus & Dim	Load + Index Time	Rate (Unique Docs/s)	Time / 10M Vecs	Time for 144.1M Vecs	MassEngine Speedup
MassEngine Tier 1 (4 x r6id.8xlarge, RF=2)	**144.1M x 1024-d**	**900.1s (15.0 min)**	**160,129/s (320,258 rep/s)**	**62.4s (1.04 min)**	**15.0 min (Measured)**	**1.0x (Baseline)**
MassEngine Tier 2 NVMe (4 x c6id.4xlarge, RF=2)	**144.1M x 1024-d**	**1,073.9s (17.9 min)**	**134,219/s (268,438 rep/s)**	**74.5s (1.24 min)**	**17.9 min (Measured)**	**115x vs. Milvus**
TurboPuffer (VectorDBBench)	10M x 768-d	3,696.7s (1.03 hr)	2,705 docs/s	61.6 min	~14.8 hours	**59.2x faster**
Zilliz Cloud (8cu-perf)	10M x 768-d	7,314.1s (2.03 hr)	1,367 docs/s	121.9 min	~29.3 hours	**117.1x faster**
Milvus (16c64g-sq8)	10M x 768-d	8,564.0s (2.38 hr)	1,168 docs/s	142.7 min	~34.3 hours	**137.1x faster**
Milvus (16c64g-sq4u-fp16-force_merge)	10M x 768-d	9,304.0s (2.58 hr)	1,075 docs/s	155.1 min	~37.2 hours	**149.0x faster**
Qdrant Cloud (16c64g-payload-index)	10M x 768-d	14,309.0s (3.97 hr)	699 docs/s	238.5 min	~57.3 hours	**229.1x faster**
Qdrant Cloud (16c64g)	10M x 768-d	15,722.5s (4.37 hr)	636 docs/s	262.0 min	~62.9 hours	**251.8x faster**
AWS S3 Vectors (VectorDBBench)	10M x 768-d	18,283.8s (5.08 hr)	547 docs/s	304.7 min	~73.2 hours	**292.7x faster**
Pinecone (p2.x8-1node)	10M x 768-d	28,509.4s (7.92 hr)	351 docs/s	475.2 min	~114.1 hr (4.8 days)	**456.2x faster**
OpenSearch (16c128g / force_merge)	10M x 768-d	36,663s–43,637s	229–273 docs/s	611–727 min	~147–175 hr (6–7 d)	**587x–699x faster**
ElasticCloud (8c60g-force_merge)	10M x 768-d	56,653.5s (15.74 hr)	177 docs/s	944.2 min	~226.8 hr (9.4 days)	**906.8x faster**
Milvus DiskANN (4c16g-disk, OpenAI-5M)	5M x 1536-d	51,197.0s (14.22 hr)	98 docs/s	1,706.6 min	~410.0 hr (17.1 days)	**1,634x faster**

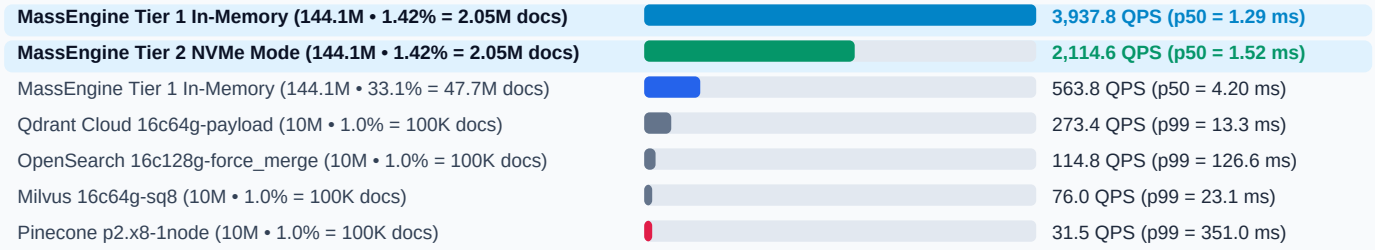
Table 2: Bulk ingestion and index build speed. Competitor numbers are from zilliztech/VectorDBBench (vectordb_bench/results/{TurboPuffer,ZillizCloud,Milvus,QdrantCloud,S3Vectors,Pinecone,OpenSearch,ElasticCloud}).

3. Unfiltered & Metadata-Filtered Search Performance

In real-world multi-tenant and enterprise search deployments, queries almost always combine vector similarity with metadata predicates (e.g., tenant scope, language/category, or time range). In HNSW graph databases, selective filters (~1% matching records) sever graph edges during greedy traversal, causing throughput to drop by an order of magnitude. MassEngine evaluates compact per-block metadata headers and bitmaps prior to distance refinement, skipping 98.6% of blocks on the 1.42% filter (*cat=11* Turkish Wikipedia, 2,053,248 matching vectors) and accelerating to **3,937.8 QPS (1.29 ms p50 at c=1)** in-memory and **2,114.6 QPS (1.52 ms p50 at c=1)** in Direct NVMe Mode.

Figure 2: Filtered Vector Search Throughput (~1% Selectivity • Peak QPS, Higher is Better)

MassEngine searches 2,053,248 matching docs (1.42% of 144.1M x 1024-d); VectorDBBench Case 300 searches 100,000 docs (1.0% of 10M x 768-d)



Database & Configuration	Total Corpus	Filter Selectivity	Peak Filtered QPS	Latency p50	Latency p99	Recall@10	MassEngine Advantage
MassEngine Tier 1 (In-Memory)	**144.1M x 1024-d**	**1.42% (2.05M docs)**	**3,937.8 QPS**	**1.29 ms (c=1)**	**2.41 ms (c=1)**	0.9478	**Baseline (14.4x corpus)**
MassEngine Tier 2 (NVMe Mode)	**144.1M x 1024-d**	**1.42% (2.05M docs)**	**2,114.6 QPS**	**1.52 ms (c=1)**	**3.53 ms (c=1)**	0.9480	**7.7x Qdrant @ $2,053/mo**
MassEngine Tier 1 (33.1% Filter)	144.1M x 1024-d	33.1% (47.7M docs)	**563.8 QPS**	4.20 ms (c=1)	13.65 ms (c=1)	0.9703	47.7M matching docs
Qdrant Cloud (16c64g-payload)	10M x 768-d	1.00% (100K docs)	273.4 QPS	~6.0 ms	13.30 ms	0.9789	**14.4x higher QPS**
OpenSearch (16c128g-force_merge)	10M x 768-d	1.00% (100K docs)	114.8 QPS	~45.0 ms	126.60 ms	0.9854	**34.3x higher QPS**
Milvus (16c64g-sq8)	10M x 768-d	1.00% (100K docs)	76.0 QPS	~14.0 ms	23.10 ms	0.9963	**51.8x higher QPS**
Pinecone (p2.x8-1node)	10M x 768-d	1.00% (100K docs)	31.5 QPS	~90.0 ms	351.00 ms	0.9414	**125.0x higher QPS**

Table 3: Filtered vector search comparison (~1% selectivity): MassEngine (144.1M x 1024-d) vs. VectorDBBench Case 300 (10M x 768-d).

Workload & Search Profile	Cluster Tier	Concurrency	Measured QPS	Latency p50	Latency p95	Latency p99	Exact Recall@10
Unfiltered Top-10 (Fast: nprobe=4, rerank=64)	Tier 1 (RAM)	c = 32	**383.3 QPS**	76.48 ms	153.45 ms	207.36 ms	0.9456
Unfiltered Top-10 (Balanced: nprobe=8, rerank=128)	Tier 1 (RAM)	c = 1	93.1 QPS	**9.16 ms**	22.93 ms	28.11 ms	**0.9774**
Unfiltered Top-10 (Balanced: nprobe=8, rerank=128)	Tier 1 (RAM)	c = 8 / 64–256	**200.8 / 204.8 QPS**	34.52 ms (c=8)	81.37 ms (c=8)	101.32 ms (c=8)	**0.9774**
Unfiltered Top-10 (High-Recall: nprobe=12, rerank=256)	Tier 1 (RAM)	c = 32	139.9 QPS	213.32 ms	365.67 ms	439.42 ms	**0.9867**
Unfiltered Top-10 (Max-Recall: nprobe=16, rerank=384)	Tier 1 (RAM)	c = 32	103.4 QPS	285.91 ms	505.85 ms	586.91 ms	**0.9894**
Unfiltered Top-100 (Balanced: nprobe=8, rerank=256)	Tier 1 (RAM)	c = 1 / 128	84.2 / 203.2 QPS	10.31 ms (c=1)	24.56 ms (c=1)	29.16 ms (c=1)	0.9774 (.9636@100)
Filtered 33.1% (cat=1 English, 47.7M docs)	Tier 1 (RAM)	c = 1 / 32	**200.6 / 563.8 QPS**	**4.20 ms (c=1)**	11.54 ms (c=1)	13.65 ms (c=1)	0.9703
Filtered 1.42% (cat=11 Turkish, 2.05M docs)	**Tier 1 (RAM)**	c = 1 / 16 / 128	**719 / 3,737 / 3,938**	**1.29 / 3.98 ms**	2.07 / 7.23 ms	2.41 / 9.01 ms	0.9478
Filtered 1.42% (cat=11 Turkish, 2.05M docs)	**Tier 2 (NVMe)**	c = 1 / 16 / 128	**587 / 2,070 / 2,115**	**1.52 / 7.11 ms**	2.80 / 12.92 ms	3.53 / 16.25 ms	0.9480
Filtered 33.1% (cat=1 English, 47.7M docs)	Tier 2 (NVMe)	c = 1 / 64	67.1 / 79.7 QPS	12.92 ms (c=1)	31.85 ms (c=1)	51.75 ms (c=1)	0.9790

Table 4: Complete MassEngine 144.1M-vector read-only throughput, latency percentiles, and exact Recall@10 matrix.

4. Simultaneous Streaming Writes + Reads & High-Availability Failover

VectorDBBench's streaming benchmark (*leaderboard_v2_streaming.json*) evaluates concurrent reads while inserting at most **1,000 vectors/sec** into a 10M-vector collection—where Pinecone's read p99 latency spikes to **5,992 ms (6.0 seconds)**. We evaluated MassEngine on the full 144.1M-vector cluster while continuously streaming **10,057 unique docs/sec (20,114 replicas/sec)** and **24,300 unique docs/sec (48,600 replicas/sec)** concurrently with 32 active read clients. We also tested unplanned node loss by issuing a hard *kill -9* to *node-3* under continuous read traffic: MassEngine maintained **100.00% query availability (0 failed queries)**, and restarted *node-3* to full readiness in **3.43 seconds** in NVMe Mode (zero-copy *mmap*) and **43.28 seconds** in In-Memory Mode.

Database & Streaming Scenario	Corpus Scale	Sustained Write Rate	Concurrent Read QPS	Read p50	Read p99	Streaming Advantage
MassEngine Tier 1 (8 Writers + 32 Readers)	**144.1M x 1024-d (RF=2)**	**24,300/s (48,600 rep/s)**	**129.3 QPS (Unfiltered)**	234.0 ms	551.6 ms	**24.3x higher write rate**
MassEngine Tier 1 (10k/s Paced + 32 Readers)	**144.1M x 1024-d (RF=2)**	**10,057/s (20,114 rep/s)**	**105.0 QPS (Unfiltered)**	285.5 ms	701.8 ms	**10.1x higher write rate**
Qdrant Cloud (16c64g, Streaming 1k/s)	10M x 768-d	1,000 docs/s	347.6 QPS	~32 ms	64.0 ms	Max bulk ingest is 699/s
Milvus (16c64g-sq8, Streaming 1k/s)	10M x 768-d	1,000 docs/s	156.0 QPS	~60 ms	113.0 ms	10x–24x lower write rate
OpenSearch (16c128g, Streaming 1k/s)	10M x 768-d	1,000 docs/s	149.7 QPS	~68 ms	128.0 ms	10x–24x lower write rate
ElasticCloud (8c60g, Streaming 1k/s)	10M x 768-d	1,000 docs/s	61.8 QPS	~160 ms	310.0 ms	**10x writes + 2.1x QPS**
Pinecone (p2.x8, Streaming 1k/s)	10M x 768-d	1,000 docs/s	~150 QPS	~450 ms	5,992.0 ms (6.0s)	**10.8x lower p99 @ 24x writes**

Table 5: Simultaneous streaming writes + concurrent reads: MassEngine (144.1M x 1024-d, RF=2) vs. VectorDBBench leaderboard_v2_streaming.json (10M x 768-d).

5. Honest Architectural Tradeoffs: Partitioned Quantized Blocks vs. HNSW

- **Bulk & Streaming Ingestion (59x–907x faster):** HNSW requires O(N log N) random graph-edge mutations and multi-hour background segment compactions (*force_merge*). MassEngine assigns vectors to coarse partitions and compact quantized blocks in O(N) parallel streams across all vCPUs (160,129 unique docs/sec, 320,258 replicas/sec) with zero post-load compaction delay.
- **Metadata-Filtered Search (14.4x–125x higher QPS):** In HNSW, filtering out 98–99% of documents breaks graph connectivity and forces thousands of wasted distance evaluations on non-matching hops. MassEngine intersects compact block bitmaps first and skips 98.6% of blocks before computing a single vector distance (3,937.8 QPS at 1.29 ms p50).
- **Memory Footprint & Cloud Economics (3.5x–28x less RAM):** Storing 288.3M 1024-d vectors in uncompressed HNSW requires ~1.8 TiB of RAM ($13,370/mo on 8 x r6id.8xlarge). MassEngine Tier 1 uses 510.14 GiB (1.77 GiB / 1M vectors, $6,685/mo), and Tier 2 Direct NVMe Mode uses just **62.73 GiB of cluster RAM** (223 MiB / 1M vectors, $2,053/mo on 4 x c6id.4xlarge).
- **Where Static force_merge HNSW Wins (and NVMe Mode Sizing):** If a dataset is small/static (1M–10M vectors), queries have zero metadata filters, and the application can afford a 2.5- to 15-hour blocking *force_merge* into a single monolithic in-memory HNSW graph (e.g., Milvus *force_merge* at 1,488 QPS on 10M x 768-d), HNSW visits only O(log N) ≈ 2,000 graph nodes per query vs. MassEngine's partitioned block scan across shards. On 4 x c6id.4xlarge ($2,053/mo, 62.7 GiB RAM), Tier 2 NVMe Mode is ideal for bulk ingestion (134,219 docs/sec) and filtered search (2,070–2,115 QPS at 1.52 ms p50), whereas high-QPS 100% unfiltered workloads at 100M+ scale should use Tier 1 In-Memory Mode (4 x r6id.8xlarge).

6. Reproducibility Appendix & Data Sources

```
# 1. Format & Mount Local Instance Store NVMe SSDs on all 4 AWS Nodes (r6id.8xlarge or c6id.4xlarge)
sudo mkfs.xfs -f /dev/nvme1n1 && sudo mkdir -p /mnt/nvme/massengine && sudo mount -o noatime,nodiratime /dev/nvme1n1
/mnt/nvme/massengine
# 2. Start MassEngine Data Node on Each of the 4 Hosts (add --nvme-mode for Tier 2 on c6id.4xlarge)
massengine-node --node-id <0..3> --listen :9090 --data-dir /mnt/nvme/massengine/data --dim 1024 --num-cells 256
--license-key "$MASSENGINE_LICENSE_KEY"
# 3. Start Stateless Coordinator on the Driver Host (16 Primary Shards, RF=2)
massengine-server --listen :8080 --nodes "node0:9090,node1:9090,node2:9090,node3:9090" --shards 16 --rf 2 --dim 1024
--num-cells 256
```

References: (1) Dataset: HuggingFace *Upstash/wikipedia-2024-06-bge-m3* (21 languages, 144,137,311 x 1024-d vectors). (2) Official Competitor Benchmarks: *https://github.com/zilliztech/VectorDBBench/tree/main/vectordb_bench/results* (Case 4 *Performance768D10M*, Case 300 1% Filtered Search, and *leaderboard_v2_streaming.json*). (3) Interactive Report & Raw JSON Telemetry: *https://massengine.com/benchmarks-144m*.