Elasticsearch Service

Product Introduction

Product Documentation

Contents

Product Introduction
Overview

Last updated：2022-06-28 21:55:27

Tencent Cloud Elasticsearch Service (ES) is a cloud-managed Elasticsearch service that is highly available and scalable. It is built on Elasticsearch, a RESTful API-based distributed search and analysis engine that can be used to search and analyze massive data.
While retaining Elasticsearch's compatibility and openness, ES incorporates Tencent Cloud's computing, storage, and security technologies. It has various cluster management functions and is secure, elastic, and highly available. Additionally, ES integrates the official Elastic Stack features (formerly X-Pack), which adds permission management, SQL, machine learning, and alert features to the open source foundation. These features simplify basic OPS tasks like cluster deployment and operation management, letting you focus on the business.
ES allows you to quickly build applications like massive data storage and search, and real-time log analysis, website search and navigation, search for enterprises, log monitoring, and click analysis.

Main Components

Elasticsearch
Elasticsearch is a distributed search engine which stores massive data, search the whole text and analyzes statistics. Its RESTful APIs and different programming language clients make it easy to develop based on your business needs.
Kibana
This is a data visualization tool that makes it easy to query and analyze the data stored in an Elasticsearch cluster.
Elastic Stack (formerly X-Pack)
Elastic Stack is Elasticsearch's official plugin which has various advanced features. Its data permission management can be applied to the field level. It can be easily integrated to your existing business via SQL and JDBC connection. Its machine learning and alerts can analyze cluster data and fluctuations to predict data trends and send out huge fluctuation alarms.

Features

Last updated：2020-02-17 14:01:08

Real-time logs of other cloud products such as CVM, CDB and TKE and the stocked and incremental business data can be aggregated and transferred to the ES cluster for distributed data storage, query and analysis.

Data collection and synchronization

You can use the Beats feature in ES to transfer data to ES cluster for storage, or to Logstash for custom conversion and parse before transferring them to ES cluster.
ES provides easy-to-use RESTful API for you to develop your own client, and you can call the data storage API to store the data in ES clusters.
ES is built in a VPC, and you can easily use various data synchronization plug-ins to sync the data of existing cloud products into ES clusters.

Data storage

ES provides different types of nodes and high-performance SSDs, ensuring the data read/write performance.
It can be elastically scaled to hundreds of nodes for data storage at the petabyte level, satisfying the needs of different business scenarios.
It can detect and replace faulty nodes, ensuring high cluster availability.
It features full-text search.

Data query, analysis and visualization

ES features full-text search, structured query, data filtering, metric statistics etc., which is applicable to information search, data analysis and many other scenarios.
ES provides easy-to-use RESTful API and clients in various languages for you to build your own search services.
With Kibana, you can easily search and statistically analyze cluster data in a browser.

Performance

Overview

Last updated：2021-08-11 11:17:22

This section describes the results of stress testing on ES instances (v7.10.1) of different specifications through the benchmark rally script officially provided by Elasticsearch.

This section provides the stress test results of ES clusters with the following specifications:

4-Core 16 GB 3-Node Cluster Performance Test

8-Core 32 GB 3-Node Cluster Performance Test

It also provides the comparison of stress test results of 4-core 16 GB and 8-core 32 GB ES clusters. For more information, please see Stress Test Result Comparison Between 4-Core 16 GB 3-Node Cluster and 8-Core 32 GB 3-Node Cluster.

4-Core 16 GB 3-Node Cluster Performance Test

Last updated：2025-02-07 14:10:00

This document describes the performance metrics of a 3-node ES cluster with 4 CPU cores, 16 GB memory, and 200 GB SSD storage capacity.

Note:

The data comes from GeoNames and contains 11,396,503 entries of geographic location data in text, long, geo, and other types stored in columns and rows with a total size of around 3 GB.

The comparison between the 4-core 16 GB SSD 200 GB 3-node ES cluster and a community edition cluster with the same specification shows that ES has better performance in all aspects thanks to its optimizations of the underlying storage model (time series merging and continuous cold shard merging), query execution plan (efficient pruning and caching), built-in scenario templates, proprietary JDK, and GC parameter tuning. For more information, please see ES Kernel Enhancement.

geonames/7.10.1/4-core 16 GB

Description	Metric	Unit	Task	ES	Com Editi
Total write time	Cumulative indexing time of primary shards	min	-	16.3633	17.8
Total GC count and time	Total Young Gen GC time	s	-	6.26	68.4
	Total Young Gen GC count	-	-	892	416
	Total Old Gen GC	s	-	0	0

	time				
	Total Old Gen GC count	-	-	0	0
Storage size	Store size	GB	-	2.51866	2.93
Heap memory usage	Heap used for segments	MB	-	0.803783	0.70
	Heap used for doc values	MB	-	0.0284767	0.02
	Heap used for terms	MB	-	0.655075	0.56
	Heap used for norms	MB	-	0.0732422	0.07
	Heap used for points	MB	-	0	0
	Heap used for stored fields	MB	-	0.0469894	0.04
Total segment count	Segment count	-	-	6	97
Write throughput and time	Min Throughput	docs/s	index-append	89331.9	806
	Median Throughput	docs/s	index-append	90268.8	825
	Max Throughput	docs/s	index-append	90516.1	840
	50th percentile latency	ms	index-append	233.258	305.
	90th percentile latency	ms	index-append	314.558	354.

	99th percentile latency	ms	index-append	341.303	403.
	100th percentile latency	ms	index-append	354.657	428.
	50th percentile service time	ms	index-append	233.258	305.
	90th percentile service time	ms	index-append	314.558	354.
	99th percentile service time	ms	index-append	341.303	403.
	100th percentile service time	ms	index-append	354.657	428.
	error rate	%	index-append	0	0
Index metrics	Min Throughput	ops/s	index-stats	90.04	90.0
	Median Throughput	ops/s	index-stats	90.07	90.0
	Max Throughput	ops/s	index-stats	90.14	90.1
	50th percentile latency	ms	index-stats	2.91003	3.07
	90th percentile latency	ms	index-stats	3.82882	4.27

	99th percentile latency	ms	index-stats	4.2378	4.75
	99.9th percentile latency	ms	index-stats	4.34459	9.23
	100th percentile latency	ms	index-stats	8.22393	17.3
	50th percentile service time	ms	index-stats	1.78268	2.17
	90th percentile service time	ms	index-stats	2.07484	2.55
	99th percentile service time	ms	index-stats	2.43121	2.94
	99.9th percentile service time	ms	index-stats	3.09198	3.50
	100th percentile service time	ms	index-stats	7.29974	15.5
	error rate	%	index-stats	0	0
Node metrics	Min Throughput	ops/s	node-stats	90.06	90.0
	Median Throughput	ops/s	node-stats	90.09	90.0
	Max Throughput	ops/s	node-stats	90.34	90.3

	50th percentile latency	ms	node-stats	3.17223	3.60
	90th percentile latency	ms	node-stats	3.70681	4.17
	99th percentile latency	ms	node-stats	5.01334	5.77
	99.9th percentile latency	ms	node-stats	6.75018	7.29
	100th percentile latency	ms	node-stats	7.98905	8.64
	50th percentile service time	ms	node-stats	2.43876	2.80
	90th percentile service time	ms	node-stats	2.78272	3.25
	99th percentile service time	ms	node-stats	4.12234	5.21
	99.9th percentile service time	ms	node-stats	6.35902	6.69
	100th percentile service time	ms	node-stats	7.4313	7.52
	error rate	%	node-stats	0	0

Default query with all documents having a score of 1 (match_all)	Min Throughput	ops/s	default	50.03	50.0
	Median Throughput	ops/s	default	50.04	50.0
	Max Throughput	ops/s	default	50.08	50.0
	50th percentile latency	ms	default	3.89929	4.97
	90th percentile latency	ms	default	4.39236	5.47
	99th percentile latency	ms	default	4.78834	6.31
	99.9th percentile latency	ms	default	7.10486	46.1
	100th percentile latency	ms	default	8.75822	59.3
	50th percentile service time	ms	default	3.18269	4.13
	90th percentile service time	ms	default	3.49347	4.47
	99th percentile service time	ms	default	3.8746	5.30
	99.9th percentile	ms	default	6.68581	9.82

	service time				
	100th percentile service time	ms	default	8.30396	58.1
	error rate	%	default	0	0
Term query	Min Throughput	ops/s	term	100.05	100.
	Median Throughput	ops/s	term	100.07	100.
	Max Throughput	ops/s	term	100.14	100.
	50th percentile latency	ms	term	3.17419	3.37
	90th percentile latency	ms	term	3.62229	3.87
	99th percentile latency	ms	term	4.03812	5.63
	99.9th percentile latency	ms	term	5.9753	7.94
	100th percentile latency	ms	term	8.03321	11.0
	50th percentile service time	ms	term	2.49755	2.59
	90th percentile service time	ms	term	2.71322	2.92

	99th percentile service time	ms	term	3.20673	4.40
	99.9th percentile service time	ms	term	5.17998	7.02
	100th percentile service time	ms	term	6.95227	10.7
	error rate	%	term	0	0
Phrase query	Min Throughput	ops/s	phrase	110.05	110.
	Median Throughput	ops/s	phrase	110.07	110.
	Max Throughput	ops/s	phrase	110.12	110.
	50th percentile latency	ms	phrase	3.09905	3.19
	90th percentile latency	ms	phrase	3.62549	3.74
	99th percentile latency	ms	phrase	4.55457	7.78
	99.9th percentile latency	ms	phrase	8.29519	20.2
	100th percentile latency	ms	phrase	9.39771	23.2
	50th	ms	phrase	2.38248	2.38

	percentile service time				
	90th percentile service time	ms	phrase	2.77084	2.75
	99th percentile service time	ms	phrase	3.75448	5.27
	99.9th percentile service time	ms	phrase	7.5974	19.2
	100th percentile service time	ms	phrase	8.98362	22.7
	error rate	%	phrase	0	0
Aggregation query without cache	Min Throughput	ops/s	country_agg_uncached	3.6	3.6
	Median Throughput	ops/s	country_agg_uncached	3.61	3.6
	Max Throughput	ops/s	country_agg_uncached	3.61	3.61
	50th percentile latency	ms	country_agg_uncached	157.466	179.
	90th percentile latency	ms	country_agg_uncached	217.148	285.
	99th percentile latency	ms	country_agg_uncached	233.185	294.
	100th	ms	country_agg_uncached	233.227	297.

	percentile latency				
	50th percentile service time	ms	country_agg_uncached	156.197	174.
	90th percentile service time	ms	country_agg_uncached	215.852	285.
	99th percentile service time	ms	country_agg_uncached	232.177	287.
	100th percentile service time	ms	country_agg_uncached	232.321	287.
	error rate	%	country_agg_uncached	0	0
Aggregation query with cache	Min Throughput	ops/s	country_agg_cached	100.03	100.
	Median Throughput	ops/s	country_agg_cached	100.05	100.
	Max Throughput	ops/s	country_agg_cached	100.08	100.
	50th percentile latency	ms	country_agg_cached	2.44457	2.43
	90th percentile latency	ms	country_agg_cached	2.97922	2.93
	99th percentile latency	ms	country_agg_cached	3.96393	3.89
	99.9th percentile	ms	country_agg_cached	5.3294	5.35

	latency				
	100th percentile latency	ms	country_agg_cached	7.9529	5.42
	50th percentile service time	ms	country_agg_cached	1.71924	1.66
	90th percentile service time	ms	country_agg_cached	1.97892	1.84
	99th percentile service time	ms	country_agg_cached	2.22611	2.17
	99.9th percentile service time	ms	country_agg_cached	5.0967	4.25
	100th percentile service time	ms	country_agg_cached	7.02246	4.95
	error rate	%	country_agg_cached	0	0
Paged pull	Min Throughput	pages/s	scroll	20.04	20.0
	Median Throughput	pages/s	scroll	20.04	20.0
	Max Throughput	pages/s	scroll	20.05	20.0
	50th percentile latency	ms	scroll	576.675	598.
	90th percentile	ms	scroll	585.156	620.

	latency				
	99th percentile latency	ms	scroll	598.95	646.
	100th percentile latency	ms	scroll	602.009	646.
	50th percentile service time	ms	scroll	575.118	596.
	90th percentile service time	ms	scroll	583.906	618.
	99th percentile service time	ms	scroll	597.482	644.
	100th percentile service time	ms	scroll	600.578	644.
	error rate	%	scroll	0	0
Script query (using expression script)	Min Throughput	ops/s	expression	2	2
	Median Throughput	ops/s	expression	2	2
	Max Throughput	ops/s	expression	2	2
	50th percentile latency	ms	expression	299.685	360.
	90th percentile latency	ms	expression	416.613	491.

	99th percentile latency	ms	expression	465.776	500.
	100th percentile latency	ms	expression	468.083	500.
	50th percentile service time	ms	expression	298.594	359.
	90th percentile service time	ms	expression	415.045	489.
	99th percentile service time	ms	expression	464.598	499.
	100th percentile service time	ms	expression	467.106	499.
	error rate	%	expression	0	0
Script query (using painless static script without dynamically getting field values)	Min Throughput	ops/s	painless_static	1.5	1.5
	Median Throughput	ops/s	painless_static	1.5	1.5
	Max Throughput	ops/s	painless_static	1.5	1.5
	50th percentile latency	ms	painless_static	383.485	389.
	90th percentile latency	ms	painless_static	514.495	641.

	99th percentile latency	ms	painless_static	561.342	644.
	100th percentile latency	ms	painless_static	568.066	646.
	50th percentile service time	ms	painless_static	382.158	388.
	90th percentile service time	ms	painless_static	513.202	640.
	99th percentile service time	ms	painless_static	560.61	642.
	100th percentile service time	ms	painless_static	567.419	644.
	error rate	%	painless_static	0	0
Script query (using painless static script with dynamically getting field values)	Min Throughput	ops/s	painless_dynamic	1.5	1.5
	Median Throughput	ops/s	painless_dynamic	1.5	1.5
	Max Throughput	ops/s	painless_dynamic	1.5	1.5
	50th percentile latency	ms	painless_dynamic	377.278	393.
	90th percentile latency	ms	painless_dynamic	517.496	633.
	99th	ms	painless_dynamic	576.697	653.

	percentile latency				
	100th percentile latency	ms	painless_dynamic	580.017	660.
	50th percentile service time	ms	painless_dynamic	376.339	391.
	90th percentile service time	ms	painless_dynamic	516.407	632.
	99th percentile service time	ms	painless_dynamic	575.714	652.
	100th percentile service time	ms	painless_dynamic	579.642	659.
	error rate	%	painless_dynamic	0	0
Geographic range query (based on Gaussian decay function)	Min Throughput	ops/s	decay_geo_gauss_function_score	1	1
	Median Throughput	ops/s	decay_geo_gauss_function_score	1	1
	Max Throughput	ops/s	decay_geo_gauss_function_score	1	1
	50th percentile latency	ms	decay_geo_gauss_function_score	348.531	388.
	90th percentile latency	ms	decay_geo_gauss_function_score	398.351	472.
	99th percentile	ms	decay_geo_gauss_function_score	411.483	492.

	latency				
	100th percentile latency	ms	decay_geo_gauss_function_score	457.615	494.
	50th percentile service time	ms	decay_geo_gauss_function_score	346.881	386.
	90th percentile service time	ms	decay_geo_gauss_function_score	397.08	470.
	99th percentile service time	ms	decay_geo_gauss_function_score	410.421	490.
	100th percentile service time	ms	decay_geo_gauss_function_score	455.704	492.
	error rate	%	decay_geo_gauss_function_score	0	0
Geographic range query (based on Gaussian decay function with dynamically getting field values through script)	Min Throughput	ops/s	decay_geo_gauss_script_score	1	1
	Median Throughput	ops/s	decay_geo_gauss_script_score	1	1
	Max Throughput	ops/s	decay_geo_gauss_script_score	1	1
	50th percentile latency	ms	decay_geo_gauss_script_score	368.275	430.
	90th percentile latency	ms	decay_geo_gauss_script_score	414.905	539.
	99th percentile latency	ms	decay_geo_gauss_script_score	468.888	543.

	100th percentile latency	ms	decay_geo_gauss_script_score	477.25	546.
	50th percentile service time	ms	decay_geo_gauss_script_score	366.945	429.
	90th percentile service time	ms	decay_geo_gauss_script_score	413.609	538.
	99th percentile service time	ms	decay_geo_gauss_script_score	467.627	542.
	100th percentile service time	ms	decay_geo_gauss_script_score	475.367	545.
	error rate	%	decay_geo_gauss_script_score	0	0
Custom scoring function query (defining function based on field value)	Min Throughput	ops/s	field_value_function_score	1.5	1.5
	Median Throughput	ops/s	field_value_function_score	1.5	1.5
	Max Throughput	ops/s	field_value_function_score	1.51	1.51
	50th percentile latency	ms	field_value_function_score	139.661	162.
	90th percentile latency	ms	field_value_function_score	183.675	215.
	99th percentile latency	ms	field_value_function_score	197.653	221.

	100th percentile latency	ms	field_value_function_score	202.345	228.
	50th percentile service time	ms	field_value_function_score	138.423	159.
	90th percentile service time	ms	field_value_function_score	182.404	214.
	99th percentile service time	ms	field_value_function_score	196.734	220.
	100th percentile service time	ms	field_value_function_score	201.442	226.
	error rate	%	field_value_function_score	0	0
Custom scoring function query (dynamically getting field values through script to calculate scores)	Min Throughput	ops/s	field_value_script_score	1.5	1.5
	Median Throughput	ops/s	field_value_script_score	1.5	1.5
	Max Throughput	ops/s	field_value_script_score	1.51	1.51
	50th percentile latency	ms	field_value_script_score	188.952	189.
	90th percentile latency	ms	field_value_script_score	264.095	313.
	99th percentile latency	ms	field_value_script_score	271.153	326.
	100th	ms	field_value_script_score	271.901	338.

	percentile latency				
	50th percentile service time	ms	field_value_script_score	187.218	187.
	90th percentile service time	ms	field_value_script_score	263.207	311.
	99th percentile service time	ms	field_value_script_score	269.578	325.
	100th percentile service time	ms	field_value_script_score	270.138	336.
	error rate	%	field_value_script_score	0	0
Large terms query	Min Throughput	ops/s	large_terms	1.1	1.1
	Median Throughput	ops/s	large_terms	1.1	1.1
	Max Throughput	ops/s	large_terms	1.1	1.1
	50th percentile latency	ms	large_terms	265.007	835.
	90th percentile latency	ms	large_terms	296.009	113
	99th percentile latency	ms	large_terms	310.358	132
	100th percentile	ms	large_terms	311.049	136

	latency				
	50th percentile service time	ms	large_terms	256.372	774.
	90th percentile service time	ms	large_terms	287.851	1022
	99th percentile service time	ms	large_terms	301.827	103
	100th percentile service time	ms	large_terms	302.251	103
	error rate	%	large_terms	0	0
Large filtered terms query	Min Throughput	ops/s	large_filtered_terms	1.1	1.1
	Median Throughput	ops/s	large_filtered_terms	1.1	1.1
	Max Throughput	ops/s	large_filtered_terms	1.1	1.1
	50th percentile latency	ms	large_filtered_terms	268.135	778.
	90th percentile latency	ms	large_filtered_terms	304.158	100
	99th percentile latency	ms	large_filtered_terms	351.209	110
	100th percentile latency	ms	large_filtered_terms	352.003	113

	50th percentile service time	ms	large_filtered_terms	259.546	695.
	90th percentile service time	ms	large_filtered_terms	295.721	997.
	99th percentile service time	ms	large_filtered_terms	342.342	102
	100th percentile service time	ms	large_filtered_terms	343.378	102
	error rate	%	large_filtered_terms	0	0
Large prohibited terms query	Min Throughput	ops/s	large_prohibited_terms	1.1	1.1
	Median Throughput	ops/s	large_prohibited_terms	1.1	1.1
	Max Throughput	ops/s	large_prohibited_terms	1.1	1.1
	50th percentile latency	ms	large_prohibited_terms	270.041	828.
	90th percentile latency	ms	large_prohibited_terms	310.351	112
	99th percentile latency	ms	large_prohibited_terms	347.414	129
	100th percentile latency	ms	large_prohibited_terms	349.499	137

	50th percentile service time	ms	large_prohibited_terms	261.734	728.
	90th percentile service time	ms	large_prohibited_terms	302.279	1012
	99th percentile service time	ms	large_prohibited_terms	339.278	1032
	100th percentile service time	ms	large_prohibited_terms	340.817	1034
	error rate	%	large_prohibited_terms	0	0
Descending order query	Min Throughput	ops/s	desc_sort_population	1.5	1.5
	Median Throughput	ops/s	desc_sort_population	1.51	1.51
	Max Throughput	ops/s	desc_sort_population	1.51	1.51
	50th percentile latency	ms	desc_sort_population	58.5828	65.9
	90th percentile latency	ms	desc_sort_population	77.9981	118.
	99th percentile latency	ms	desc_sort_population	80.8863	119.
	100th percentile latency	ms	desc_sort_population	83.1661	119.
	50th	ms	desc_sort_population	57.1212	64.2

	percentile service time				
	90th percentile service time	ms	desc_sort_population	76.7082	117.
	99th percentile service time	ms	desc_sort_population	79.2907	117.
	100th percentile service time	ms	desc_sort_population	81.6364	117.
	error rate	%	desc_sort_population	0	0
Ascending order query	Min Throughput	ops/s	asc_sort_population	1.5	1.5
	Median Throughput	ops/s	asc_sort_population	1.51	1.51
	Max Throughput	ops/s	asc_sort_population	1.51	1.51
	50th percentile latency	ms	asc_sort_population	62.4328	86.1
	90th percentile latency	ms	asc_sort_population	79.8441	123.
	99th percentile latency	ms	asc_sort_population	83.9411	124.
	100th percentile latency	ms	asc_sort_population	84.3925	125.
	50th percentile	ms	asc_sort_population	61.0637	84.9

	service time				
	90th percentile service time	ms	asc_sort_population	78.4101	122.
	99th percentile service time	ms	asc_sort_population	82.2652	123.
	100th percentile service time	ms	asc_sort_population	82.5616	124.
	error rate	%	asc_sort_population	0	0
search_after query with sorting in ascending order	Min Throughput	ops/s	asc_sort_with_after_population	1.5	1.5
	Median Throughput	ops/s	asc_sort_with_after_population	1.51	1.5
	Max Throughput	ops/s	asc_sort_with_after_population	1.51	1.51
	50th percentile latency	ms	asc_sort_with_after_population	88.1871	99.9
	90th percentile latency	ms	asc_sort_with_after_population	127.995	173.
	99th percentile latency	ms	asc_sort_with_after_population	131.171	174.
	100th percentile latency	ms	asc_sort_with_after_population	132.181	174.
	50th percentile	ms	asc_sort_with_after_population	87.132	98.2

	service time				
	90th percentile service time	ms	asc_sort_with_after_population	126.818	171.
	99th percentile service time	ms	asc_sort_with_after_population	129.453	171.
	100th percentile service time	ms	asc_sort_with_after_population	130.452	171.
	error rate	%	asc_sort_with_after_population	0	0
Query with sorting high base fields in descending order (quickly getting topK based on DistanceFeatureQuery)	Min Throughput	ops/s	desc_sort_geonameid	6.02	6.02
	Median Throughput	ops/s	desc_sort_geonameid	6.02	6.02
	Max Throughput	ops/s	desc_sort_geonameid	6.03	6.03
	50th percentile latency	ms	desc_sort_geonameid	7.4659	7.55
	90th percentile latency	ms	desc_sort_geonameid	8.26766	9.07
	99th percentile latency	ms	desc_sort_geonameid	8.72369	9.69
	100th percentile latency	ms	desc_sort_geonameid	8.79956	10.4
	50th percentile	ms	desc_sort_geonameid	6.59986	6.52

	service time				
	90th percentile service time	ms	desc_sort_geonameid	7.24539	7.85
	99th percentile service time	ms	desc_sort_geonameid	7.57925	8.40
	100th percentile service time	ms	desc_sort_geonameid	7.64471	9.40
	error rate	%	desc_sort_geonameid	0	0
search_after query with sorting high base fields in descending order	Min Throughput	ops/s	desc_sort_with_after_geonameid	6.01	6
	Median Throughput	ops/s	desc_sort_with_after_geonameid	6.01	6.01
	Max Throughput	ops/s	desc_sort_with_after_geonameid	6.02	6.01
	50th percentile latency	ms	desc_sort_with_after_geonameid	89.4587	107.
	90th percentile latency	ms	desc_sort_with_after_geonameid	119.777	154.
	99th percentile latency	ms	desc_sort_with_after_geonameid	123.271	155.
	100th percentile latency	ms	desc_sort_with_after_geonameid	123.628	156.
	50th percentile	ms	desc_sort_with_after_geonameid	88.512	107.

	service time				
	90th percentile service time	ms	desc_sort_with_after_geonameid	118.72	153.
	99th percentile service time	ms	desc_sort_with_after_geonameid	122.79	153.
	100th percentile service time	ms	desc_sort_with_after_geonameid	122.791	154.
	error rate	%	desc_sort_with_after_geonameid	0	0
Query with sorting high base fields in ascending order (quickly getting topK based on DistanceFeatureQuery)	Min Throughput	ops/s	asc_sort_geonameid	6.02	6.02
	Median Throughput	ops/s	asc_sort_geonameid	6.02	6.02
	Max Throughput	ops/s	asc_sort_geonameid	6.03	6.03
	50th percentile latency	ms	asc_sort_geonameid	5.80593	5.78
	90th percentile latency	ms	asc_sort_geonameid	6.55438	6.60
	99th percentile latency	ms	asc_sort_geonameid	7.36432	8.35
	100th percentile latency	ms	asc_sort_geonameid	7.49672	30.8
	50th percentile	ms	asc_sort_geonameid	4.91916	4.95

	service time				
	90th percentile service time	ms	asc_sort_geonameid	5.61126	5.26
	99th percentile service time	ms	asc_sort_geonameid	6.12285	7.42
	100th percentile service time	ms	asc_sort_geonameid	6.51222	29.8
	error rate	%	asc_sort_geonameid	0	0
search_after query with sorting high base fields in ascending order	Min Throughput	ops/s	asc_sort_with_after_geonameid	6.01	6
	Median Throughput	ops/s	asc_sort_with_after_geonameid	6.01	6.01
	Max Throughput	ops/s	asc_sort_with_after_geonameid	6.01	6.02
	50th percentile latency	ms	asc_sort_with_after_geonameid	70.994	102.
	90th percentile latency	ms	asc_sort_with_after_geonameid	104.817	137.
	99th percentile latency	ms	asc_sort_with_after_geonameid	108.797	139.
	100th percentile latency	ms	asc_sort_with_after_geonameid	108.929	140.
	50th percentile	ms	asc_sort_with_after_geonameid	69.7056	101.

	service time				
	90th percentile service time	ms	asc_sort_with_after_geonameid	103.875	136.
	99th percentile service time	ms	asc_sort_with_after_geonameid	107.828	138.
	100th percentile service time	ms	asc_sort_with_after_geonameid	108.539	139.
	error rate	%	asc_sort_with_after_geonameid	0	0

8-Core 32 GB 3-Node Cluster Performance Test

Last updated：2024-11-29 18:00:33

This document describes the performance metrics of a 3-node ES cluster with 8 CPU cores and 32 GB memory, and 200 GB SSD storage capacity.

Note:

The data comes from GeoNames and contains 11,396,503 entries of geographic location data in text, long, geo, and other types stored in columns and rows with a total size of around 3 GB.

The comparison between the 8-core 32 GB SSD 200 GB 3-node ES cluster and a community edition cluster with the same specification shows that ES has better performance in all aspects thanks to its optimizations of the underlying storage model (time series merging and continuous cold shard merging), query execution plan (efficient pruning and caching), built-in scenario templates, proprietary JDK, and GC parameter tuning. For more information, please see ES Kernel Enhancement.

geonames/7.10.1/8-core 32 GB

Description	Metric	Unit	Task	ES	Com Editi
Total write time	Cumulative indexing time of primary shards	min	-	14.2567	15.4
Total GC count and time	Total Young Gen GC time	s	-	3.544	17.7
	Total Young Gen GC count	-	-	447	108
	Total Old Gen GC	s	-	0	0

	time				
	Total Old Gen GC count	-	-	0	0
Storage size	Store size	GB	-	2.59725	3.07
Heap memory usage	Heap used for segments	MB	-	0.534325	0.76
	Heap used for doc values	MB	-	0.0507355	0.03
	Heap used for terms	MB	-	0.370026	0.60
	Heap used for norms	MB	-	0.0396729	0.08
	Heap used for points	MB	-	0	0
	Heap used for stored fields	MB	-	0.0119553	0.05
Total segment count	Segment count	-	-	7	105
Write throughput and time	Min Throughput	docs/s	index-append	153730	132
	Median Throughput	docs/s	index-append	159765	142
	Max Throughput	docs/s	index-append	162791	148
	50th percentile latency	ms	index-append	130.877	151.
	90th percentile latency	ms	index-append	162.969	187.

	99th percentile latency	ms	index-append	181.428	240.
	100th percentile latency	ms	index-append	225.98	285.
	50th percentile service time	ms	index-append	130.877	151.
	90th percentile service time	ms	index-append	162.969	187.
	99th percentile service time	ms	index-append	181.428	240.
	100th percentile service time	ms	index-append	225.98	285.
	error rate	%	index-append	0	0
Index metrics	Min Throughput	ops/s	index-stats	90.05	90.0
	Median Throughput	ops/s	index-stats	90.06	90.0
	Max Throughput	ops/s	index-stats	90.12	90.1
	50th percentile latency	ms	index-stats	2.76736	2.65
	90th percentile latency	ms	index-stats	3.58235	3.48

	99th percentile latency	ms	index-stats	3.95798	3.89
	99.9th percentile latency	ms	index-stats	4.39377	9.07
	100th percentile latency	ms	index-stats	9.00375	18.1
	50th percentile service time	ms	index-stats	1.57744	1.45
	90th percentile service time	ms	index-stats	1.8317	1.66
	99th percentile service time	ms	index-stats	2.0752	1.94
	99.9th percentile service time	ms	index-stats	2.24891	2.64
	100th percentile service time	ms	index-stats	2.31078	16.5
	error rate	%	index-stats	0	0
Node metrics	Min Throughput	ops/s	node-stats	90.06	90.0
	Median Throughput	ops/s	node-stats	90.12	90.1
	Max Throughput	ops/s	node-stats	90.36	90.3

	50th percentile latency	ms	node-stats	2.9754	2.93
	90th percentile latency	ms	node-stats	4.07929	4.02
	99th percentile latency	ms	node-stats	5.0754	4.78
	99.9th percentile latency	ms	node-stats	6.53613	17.0
	100th percentile latency	ms	node-stats	6.93454	24.7
	50th percentile service time	ms	node-stats	2.23841	2.17
	90th percentile service time	ms	node-stats	2.65367	2.62
	99th percentile service time	ms	node-stats	3.92073	3.93
	99.9th percentile service time	ms	node-stats	4.92842	4.48
	100th percentile service time	ms	node-stats	5.92757	23.7
	error rate	%	node-stats	0	0

Default query with all documents having a score of 1 (match_all)	Min Throughput	ops/s	default	50.03	50.0
	Median Throughput	ops/s	default	50.04	50.0
	Max Throughput	ops/s	default	50.08	50.0
	50th percentile latency	ms	default	3.53894	3.40
	90th percentile latency	ms	default	4.11403	4.57
	99th percentile latency	ms	default	4.92737	5.37
	99.9th percentile latency	ms	default	5.74037	25.8
	100th percentile latency	ms	default	7.32557	27.2
	50th percentile service time	ms	default	2.7831	2.67
	90th percentile service time	ms	default	3.17322	2.97
	99th percentile service time	ms	default	3.77477	3.38
	99.9th percentile	ms	default	4.19186	25.1

	service time				
	100th percentile service time	ms	default	6.58243	26.1
	error rate	%	default	0	0
Term query	Min Throughput	ops/s	term	99.66	100.
	Median Throughput	ops/s	term	100.07	100.
	Max Throughput	ops/s	term	100.11	100.
	50th percentile latency	ms	term	2.83987	2.72
	90th percentile latency	ms	term	3.32569	3.16
	99th percentile latency	ms	term	3.96055	3.66
	99.9th percentile latency	ms	term	4.33961	9.62
	100th percentile latency	ms	term	5.70421	16.4
	50th percentile service time	ms	term	2.08935	2.00
	90th percentile service time	ms	term	2.53284	2.24

	99th percentile service time	ms	term	2.99484	2.64
	99.9th percentile service time	ms	term	3.37709	6.10
	100th percentile service time	ms	term	5.24029	15.8
	error rate	%	term	0	0
Phrase query	Min Throughput	ops/s	phrase	110.04	110.
	Median Throughput	ops/s	phrase	110.08	110.
	Max Throughput	ops/s	phrase	110.11	110.
	50th percentile latency	ms	phrase	2.74088	2.89
	90th percentile latency	ms	phrase	3.30207	3.39
	99th percentile latency	ms	phrase	4.8127	8.22
	99.9th percentile latency	ms	phrase	5.57204	25.9
	100th percentile latency	ms	phrase	6.54587	27.9
	50th	ms	phrase	1.98839	2.18

	percentile service time				
	90th percentile service time	ms	phrase	2.41365	2.51
	99th percentile service time	ms	phrase	4.00121	3.24
	99.9th percentile service time	ms	phrase	4.70793	25.7
	100th percentile service time	ms	phrase	5.67829	27.4
	error rate	%	phrase	0	0
Aggregation query without cache	Min Throughput	ops/s	country_agg_uncached	3.6	3.6
	Median Throughput	ops/s	country_agg_uncached	3.61	3.61
	Max Throughput	ops/s	country_agg_uncached	3.61	3.61
	50th percentile latency	ms	country_agg_uncached	130.314	162.
	90th percentile latency	ms	country_agg_uncached	147.567	176.
	99th percentile latency	ms	country_agg_uncached	165.174	184.
	100th	ms	country_agg_uncached	174.015	269.

	percentile latency				
	50th percentile service time	ms	country_agg_uncached	129.186	161.
	90th percentile service time	ms	country_agg_uncached	146.921	175.
	99th percentile service time	ms	country_agg_uncached	164.579	183.
	100th percentile service time	ms	country_agg_uncached	172.827	269.
	error rate	%	country_agg_uncached	0	0
Aggregation query with cache	Min Throughput	ops/s	country_agg_cached	100.04	100.
	Median Throughput	ops/s	country_agg_cached	100.05	100.
	Max Throughput	ops/s	country_agg_cached	100.08	100.
	50th percentile latency	ms	country_agg_cached	2.29531	2.10
	90th percentile latency	ms	country_agg_cached	3.57418	3.49
	99th percentile latency	ms	country_agg_cached	3.91685	3.74
	99.9th percentile	ms	country_agg_cached	4.19749	4.05

	latency				
	100th percentile latency	ms	country_agg_cached	4.51842	4.63
	50th percentile service time	ms	country_agg_cached	1.57861	1.40
	90th percentile service time	ms	country_agg_cached	1.89111	1.58
	99th percentile service time	ms	country_agg_cached	2.19488	1.74
	99.9th percentile service time	ms	country_agg_cached	3.42563	3.46
	100th percentile service time	ms	country_agg_cached	4.28971	3.97
	error rate	%	country_agg_cached	0	0
Paged pull	Min Throughput	pages/s	scroll	20.04	20.0
	Median Throughput	pages/s	scroll	20.05	20.0
	Max Throughput	pages/s	scroll	20.06	20.0
	50th percentile latency	ms	scroll	538.421	556.
	90th percentile	ms	scroll	543.566	573.

	latency				
	99th percentile latency	ms	scroll	582.263	585.
	100th percentile latency	ms	scroll	584.75	587.
	50th percentile service time	ms	scroll	537.068	554.
	90th percentile service time	ms	scroll	542.428	572.
	99th percentile service time	ms	scroll	580.372	583.
	100th percentile service time	ms	scroll	583.612	584.
	error rate	%	scroll	0	0
Script query (using expression script)	Min Throughput	ops/s	expression	2	2
	Median Throughput	ops/s	expression	2	2
	Max Throughput	ops/s	expression	2	2
	50th percentile latency	ms	expression	265.631	277.
	90th percentile latency	ms	expression	287.121	299.

	99th percentile latency	ms	expression	311.788	434.
	100th percentile latency	ms	expression	391.745	446.
	50th percentile service time	ms	expression	264.462	276.
	90th percentile service time	ms	expression	285.113	298.
	99th percentile service time	ms	expression	310.991	434.
	100th percentile service time	ms	expression	390.33	445.
	error rate	%	expression	0	0
Script query (using painless static script without dynamically getting field values)	Min Throughput	ops/s	painless_static	1.5	1.5
	Median Throughput	ops/s	painless_static	1.5	1.5
	Max Throughput	ops/s	painless_static	1.5	1.5
	50th percentile latency	ms	painless_static	337.96	364.
	90th percentile latency	ms	painless_static	358.738	383.

	99th percentile latency	ms	painless_static	375.017	459.
	100th percentile latency	ms	painless_static	395.417	557.
	50th percentile service time	ms	painless_static	337.111	363.
	90th percentile service time	ms	painless_static	357.771	382.
	99th percentile service time	ms	painless_static	374.121	458.
	100th percentile service time	ms	painless_static	394.632	556.
	error rate	%	painless_static	0	0
Script query (using painless static script with dynamically getting field values)	Min Throughput	ops/s	painless_dynamic	1.5	1.5
	Median Throughput	ops/s	painless_dynamic	1.5	1.5
	Max Throughput	ops/s	painless_dynamic	1.5	1.5
	50th percentile latency	ms	painless_dynamic	334.684	365.
	90th percentile latency	ms	painless_dynamic	354.406	390.
	99th	ms	painless_dynamic	377.214	552.

	percentile latency				
	100th percentile latency	ms	painless_dynamic	381.276	554.
	50th percentile service time	ms	painless_dynamic	333.654	364.
	90th percentile service time	ms	painless_dynamic	353.246	389.
	99th percentile service time	ms	painless_dynamic	375.956	551.
	100th percentile service time	ms	painless_dynamic	379.75	553.
	error rate	%	painless_dynamic	0	0
Geographic range query (based on Gaussian decay function)	Min Throughput	ops/s	decay_geo_gauss_function_score	1	1
	Median Throughput	ops/s	decay_geo_gauss_function_score	1	1
	Max Throughput	ops/s	decay_geo_gauss_function_score	1	1
	50th percentile latency	ms	decay_geo_gauss_function_score	327.972	327.
	90th percentile latency	ms	decay_geo_gauss_function_score	336.979	344.
	99th percentile	ms	decay_geo_gauss_function_score	343.562	417.

	latency				
	100th percentile latency	ms	decay_geo_gauss_function_score	344.135	420.
	50th percentile service time	ms	decay_geo_gauss_function_score	326.554	326.
	90th percentile service time	ms	decay_geo_gauss_function_score	336.053	343.
	99th percentile service time	ms	decay_geo_gauss_function_score	342.151	416.
	100th percentile service time	ms	decay_geo_gauss_function_score	342.843	419.
	error rate	%	decay_geo_gauss_function_score	0	0
Geographic range query (based on Gaussian decay function with dynamically getting field values through script)	Min Throughput	ops/s	decay_geo_gauss_script_score	1	1
	Median Throughput	ops/s	decay_geo_gauss_script_score	1	1
	Max Throughput	ops/s	decay_geo_gauss_script_score	1	1
	50th percentile latency	ms	decay_geo_gauss_script_score	341.152	344.
	90th percentile latency	ms	decay_geo_gauss_script_score	349.94	371.
	99th percentile latency	ms	decay_geo_gauss_script_score	354.76	420.

	100th percentile latency	ms	decay_geo_gauss_script_score	364.169	438.
	50th percentile service time	ms	decay_geo_gauss_script_score	339.967	342.
	90th percentile service time	ms	decay_geo_gauss_script_score	348.493	370.
	99th percentile service time	ms	decay_geo_gauss_script_score	353.559	418.
	100th percentile service time	ms	decay_geo_gauss_script_score	362.748	437.
	error rate	%	decay_geo_gauss_script_score	0	0
Custom scoring function query (defining function based on field value)	Min Throughput	ops/s	field_value_function_score	1.5	1.5
	Median Throughput	ops/s	field_value_function_score	1.5	1.5
	Max Throughput	ops/s	field_value_function_score	1.51	1.51
	50th percentile latency	ms	field_value_function_score	120.538	129.
	90th percentile latency	ms	field_value_function_score	137.702	152.
	99th percentile latency	ms	field_value_function_score	147.851	185.

	100th percentile latency	ms	field_value_function_score	169.961	186.
	50th percentile service time	ms	field_value_function_score	119.159	128.
	90th percentile service time	ms	field_value_function_score	136.338	151.
	99th percentile service time	ms	field_value_function_score	146.981	184.
	100th percentile service time	ms	field_value_function_score	168.964	185.
	error rate	%	field_value_function_score	0	0
Custom scoring function query (dynamically getting field values through script to calculate scores)	Min Throughput	ops/s	field_value_script_score	1.5	1.5
	Median Throughput	ops/s	field_value_script_score	1.5	1.5
	Max Throughput	ops/s	field_value_script_score	1.51	1.51
	50th percentile latency	ms	field_value_script_score	168.069	171.
	90th percentile latency	ms	field_value_script_score	178.933	184.
	99th percentile latency	ms	field_value_script_score	196.982	200.
	100th	ms	field_value_script_score	198.722	206.

	percentile latency				
	50th percentile service time	ms	field_value_script_score	166.827	170.
	90th percentile service time	ms	field_value_script_score	177.869	183.
	99th percentile service time	ms	field_value_script_score	195.586	199.
	100th percentile service time	ms	field_value_script_score	197.054	205.
	error rate	%	field_value_script_score	0	0
Large terms query	Min Throughput	ops/s	large_terms	1.1	1.1
	Median Throughput	ops/s	large_terms	1.1	1.1
	Max Throughput	ops/s	large_terms	1.1	1.1
	50th percentile latency	ms	large_terms	241.322	597.
	90th percentile latency	ms	large_terms	252.637	600.
	99th percentile latency	ms	large_terms	265.807	749.
	100th percentile	ms	large_terms	272.611	751.

	latency				
	50th percentile service time	ms	large_terms	233.129	589.
	90th percentile service time	ms	large_terms	244.494	593.
	99th percentile service time	ms	large_terms	258.894	742.
	100th percentile service time	ms	large_terms	264.352	743.
	error rate	%	large_terms	0	0
Large filtered terms query	Min Throughput	ops/s	large_filtered_terms	1.1	1.1
	Median Throughput	ops/s	large_filtered_terms	1.1	1.1
	Max Throughput	ops/s	large_filtered_terms	1.1	1.1
	50th percentile latency	ms	large_filtered_terms	233.192	596.
	90th percentile latency	ms	large_filtered_terms	241.102	604.
	99th percentile latency	ms	large_filtered_terms	251.835	717.
	100th percentile latency	ms	large_filtered_terms	260.27	749.

	50th percentile service time	ms	large_filtered_terms	225.052	588.
	90th percentile service time	ms	large_filtered_terms	233.16	595.
	99th percentile service time	ms	large_filtered_terms	243.603	708.
	100th percentile service time	ms	large_filtered_terms	252.129	741.
	error rate	%	large_filtered_terms	0	0
Large prohibited terms query	Min Throughput	ops/s	large_prohibited_terms	1.1	1.1
	Median Throughput	ops/s	large_prohibited_terms	1.1	1.1
	Max Throughput	ops/s	large_prohibited_terms	1.1	1.1
	50th percentile latency	ms	large_prohibited_terms	235.179	610.
	90th percentile latency	ms	large_prohibited_terms	241.076	631.
	99th percentile latency	ms	large_prohibited_terms	255.983	774.
	100th percentile latency	ms	large_prohibited_terms	259.046	779.

	50th percentile service time	ms	large_prohibited_terms	227.487	603.
	90th percentile service time	ms	large_prohibited_terms	233.792	624.
	99th percentile service time	ms	large_prohibited_terms	248.53	767.
	100th percentile service time	ms	large_prohibited_terms	251.083	771.
	error rate	%	large_prohibited_terms	0	0
Descending order query	Min Throughput	ops/s	desc_sort_population	1.5	1.5
	Median Throughput	ops/s	desc_sort_population	1.51	1.51
	Max Throughput	ops/s	desc_sort_population	1.51	1.51
	50th percentile latency	ms	desc_sort_population	48.387	56.6
	90th percentile latency	ms	desc_sort_population	63.073	94.9
	99th percentile latency	ms	desc_sort_population	71.7498	97.2
	100th percentile latency	ms	desc_sort_population	83.3593	97.7
	50th	ms	desc_sort_population	47.0436	55.1

	percentile service time				
	90th percentile service time	ms	desc_sort_population	61.3731	93.9
	99th percentile service time	ms	desc_sort_population	70.4811	95.6
	100th percentile service time	ms	desc_sort_population	81.6517	96.0
	error rate	%	desc_sort_population	0	0
Ascending order query	Min Throughput	ops/s	asc_sort_population	1.5	1.5
	Median Throughput	ops/s	asc_sort_population	1.51	1.51
	Max Throughput	ops/s	asc_sort_population	1.51	1.51
	50th percentile latency	ms	asc_sort_population	49.2469	57.9
	90th percentile latency	ms	asc_sort_population	67.5894	76.0
	99th percentile latency	ms	asc_sort_population	84.6384	97.5
	100th percentile latency	ms	asc_sort_population	85.8124	97.5
	50th percentile	ms	asc_sort_population	47.8438	56.5

	service time				
	90th percentile service time	ms	asc_sort_population	66.0821	74.7
	99th percentile service time	ms	asc_sort_population	83.6026	96.2
	100th percentile service time	ms	asc_sort_population	84.2175	96.6
	error rate	%	asc_sort_population	0	0
search_after query with sorting in ascending order	Min Throughput	ops/s	asc_sort_with_after_population	1.5	1.5
	Median Throughput	ops/s	asc_sort_with_after_population	1.51	1.51
	Max Throughput	ops/s	asc_sort_with_after_population	1.51	1.51
	50th percentile latency	ms	asc_sort_with_after_population	99.1943	83.4
	90th percentile latency	ms	asc_sort_with_after_population	86.0298	98.8
	99th percentile latency	ms	asc_sort_with_after_population	102.268	131.
	100th percentile latency	ms	asc_sort_with_after_population	106.33	132.
	50th percentile	ms	asc_sort_with_after_population	68.2272	82.1

	service time				
	90th percentile service time	ms	asc_sort_with_after_population	84.685	97.2
	99th percentile service time	ms	asc_sort_with_after_population	101.133	130.
	100th percentile service time	ms	asc_sort_with_after_population	105.094	131.
	error rate	%	asc_sort_with_after_population	0	0
Query with sorting high base fields in descending order (quickly getting topK based on DistanceFeatureQuery)	Min Throughput	ops/s	desc_sort_geonameid	6.02	6.02
	Median Throughput	ops/s	desc_sort_geonameid	6.02	6.02
	Max Throughput	ops/s	desc_sort_geonameid	6.03	6.03
	50th percentile latency	ms	desc_sort_geonameid	5.53008	5.15
	90th percentile latency	ms	desc_sort_geonameid	6.20276	6.05
	99th percentile latency	ms	desc_sort_geonameid	6.67673	7.41
	100th percentile latency	ms	desc_sort_geonameid	6.95103	24.1
	50th percentile	ms	desc_sort_geonameid	4.61231	4.38

	service time				
	90th percentile service time	ms	desc_sort_geonameid	5.45982	5.20
	99th percentile service time	ms	desc_sort_geonameid	5.65304	5.75
	100th percentile service time	ms	desc_sort_geonameid	5.65578	23.3
	error rate	%	desc_sort_geonameid	0	0
search_after query with sorting high base fields in descending order	Min Throughput	ops/s	desc_sort_with_after_geonameid	6.01	6.01
	Median Throughput	ops/s	desc_sort_with_after_geonameid	6.02	6.01
	Max Throughput	ops/s	desc_sort_with_after_geonameid	6.02	6.02
	50th percentile latency	ms	desc_sort_with_after_geonameid	56.5947	75.9
	90th percentile latency	ms	desc_sort_with_after_geonameid	79.6503	88.6
	99th percentile latency	ms	desc_sort_with_after_geonameid	87.7773	117.
	100th percentile latency	ms	desc_sort_with_after_geonameid	89.3947	118.
	50th percentile	ms	desc_sort_with_after_geonameid	55.4855	75.1

	service time				
	90th percentile service time	ms	desc_sort_with_after_geonameid	79.2349	87.7
	99th percentile service time	ms	desc_sort_with_after_geonameid	87.3803	116.
	100th percentile service time	ms	desc_sort_with_after_geonameid	88.3606	117.
	error rate	%	desc_sort_with_after_geonameid	0	0
Query with sorting high base fields in ascending order (quickly getting topK based on DistanceFeatureQuery)	Min Throughput	ops/s	asc_sort_geonameid	6.02	6.02
	Median Throughput	ops/s	asc_sort_geonameid	6.02	6.02
	Max Throughput	ops/s	asc_sort_geonameid	6.03	6.03
	50th percentile latency	ms	asc_sort_geonameid	5.19317	4.49
	90th percentile latency	ms	asc_sort_geonameid	5.74438	5.01
	99th percentile latency	ms	asc_sort_geonameid	6.22846	5.49
	100th percentile latency	ms	asc_sort_geonameid	11.6377	5.53
	50th percentile	ms	asc_sort_geonameid	4.35586	3.56

	service time				
	90th percentile service time	ms	asc_sort_geonameid	4.92152	3.97
	99th percentile service time	ms	asc_sort_geonameid	5.38949	4.33
	100th percentile service time	ms	asc_sort_geonameid	10.6436	4.56
	error rate	%	asc_sort_geonameid	0	0
search_after query with sorting high base fields in ascending order	Min Throughput	ops/s	asc_sort_with_after_geonameid	6.01	6.01
	Median Throughput	ops/s	asc_sort_with_after_geonameid	6.02	6.01
	Max Throughput	ops/s	asc_sort_with_after_geonameid	6.02	6.02
	50th percentile latency	ms	asc_sort_with_after_geonameid	58.1403	69.5
	90th percentile latency	ms	asc_sort_with_after_geonameid	76.5695	81.7
	99th percentile latency	ms	asc_sort_with_after_geonameid	91.6296	98.1
	100th percentile latency	ms	asc_sort_with_after_geonameid	91.6364	104.
	50th percentile	ms	asc_sort_with_after_geonameid	57.1683	68.5

	service time				
	90th percentile service time	ms	asc_sort_with_after_geonameid	75.7573	81.0
	99th percentile service time	ms	asc_sort_with_after_geonameid	91.1533	97.8
	100th percentile service time	ms	asc_sort_with_after_geonameid	91.3662	103.
	error rate	%	asc_sort_with_after_geonameid	0	0

Stress Test Result Comparison Between 4-Core 16 GB 3-Node Cluster and 8-Core 32 GB 3-Node Cluster

Last updated：2024-11-29 18:53:25

This document compares the stress test result between a 3-node ES cluster with 4 CPU cores and 16 GB memory 200 GB SSD storage capacity and one with 8 CPU cores and 32 GB memory 200 GB SSD storage capacity.

Note:

The data comes from GeoNames and contains 11,396,503 entries of geographic location data in text, long, geo, and other types stored in columns and rows with a total size of around 3 GB.

Description	Metric	Unit	Task	4-core 16 GB	8-co GB
Total write time	Cumulative indexing time of primary shards	min	-	16.3633	14.2
Total GC count and time	Total Young Gen GC time	s	-	6.26	3.54
	Total Young Gen GC count	-	-	892	447
	Total Old Gen GC time	s	-	0	0
	Total Old Gen GC count	-	-	0	0
Storage size	Store size	GB	-	2.51866	2.59
Heap memory usage	Heap used for segments	MB	-	0.803783	0.53
	Heap used	MB	-	0.0284767	0.05

	for doc values				
	Heap used for terms	MB	-	0.655075	0.37
	Heap used for norms	MB	-	0.0732422	0.03
	Heap used for points	MB	-	0	0
	Heap used for stored fields	MB	-	0.0469894	0.01
Total segment count	Segment count	-	-	6	7
Write throughput and time	Min Throughput	docs/s	index-append	89331.9	153
	Median Throughput	docs/s	index-append	90268.8	159
	Max Throughput	docs/s	index-append	90516.1	162
	50th percentile latency	ms	index-append	233.258	130.
	90th percentile latency	ms	index-append	314.558	162.
	99th percentile latency	ms	index-append	341.303	181.
	100th percentile latency	ms	index-append	354.657	225.
	50th percentile service time	ms	index-append	233.258	130.

	90th percentile service time	ms	index-append	314.558	162.
	99th percentile service time	ms	index-append	341.303	181.
	100th percentile service time	ms	index-append	354.657	225.
	error rate	%	index-append	0	0
Index metrics	Min Throughput	ops/s	index-stats	90.04	90.0
	Median Throughput	ops/s	index-stats	90.07	90.0
	Max Throughput	ops/s	index-stats	90.14	90.1
	50th percentile latency	ms	index-stats	2.91003	2.76
	90th percentile latency	ms	index-stats	3.82882	3.58
	99th percentile latency	ms	index-stats	4.2378	3.95
	99.9th percentile latency	ms	index-stats	4.34459	4.39
	100th percentile latency	ms	index-stats	8.22393	9.00
	50th	ms	index-stats	1.78268	1.57

	percentile service time				
	90th percentile service time	ms	index-stats	2.07484	1.83
	99th percentile service time	ms	index-stats	2.43121	2.07
	99.9th percentile service time	ms	index-stats	3.09198	2.24
	100th percentile service time	ms	index-stats	7.29974	2.31
	error rate	%	index-stats	0	0
Node metrics	Min Throughput	ops/s	node-stats	90.06	90.0
	Median Throughput	ops/s	node-stats	90.09	90.1
	Max Throughput	ops/s	node-stats	90.34	90.3
	50th percentile latency	ms	node-stats	3.17223	2.97
	90th percentile latency	ms	node-stats	3.70681	4.07
	99th percentile latency	ms	node-stats	5.01334	5.07
	99.9th	ms	node-stats	6.75018	6.53

	percentile latency				
	100th percentile latency	ms	node-stats	7.98905	6.93
	50th percentile service time	ms	node-stats	2.43876	2.23
	90th percentile service time	ms	node-stats	2.78272	2.65
	99th percentile service time	ms	node-stats	4.12234	3.92
	99.9th percentile service time	ms	node-stats	6.35902	4.92
	100th percentile service time	ms	node-stats	7.4313	5.92
	error rate	%	node-stats	0	0
Default query with all documents having a score of 1 (match_all)	Min Throughput	ops/s	default	50.03	50.0
	Median Throughput	ops/s	default	50.04	50.0
	Max Throughput	ops/s	default	50.08	50.0
	50th percentile latency	ms	default	3.89929	3.53
	90th	ms	default	4.39236	4.11

	percentile latency				
	99th percentile latency	ms	default	4.78834	4.92
	99.9th percentile latency	ms	default	7.10486	5.74
	100th percentile latency	ms	default	8.75822	7.32
	50th percentile service time	ms	default	3.18269	2.78
	90th percentile service time	ms	default	3.49347	3.17
	99th percentile service time	ms	default	3.8746	3.77
	99.9th percentile service time	ms	default	6.68581	4.19
	100th percentile service time	ms	default	8.30396	6.58
	error rate	%	default	0	0
Term query	Min Throughput	ops/s	term	100.05	99.6
	Median Throughput	ops/s	term	100.07	100.

	Max Throughput	ops/s	term	100.14	100.
	50th percentile latency	ms	term	3.17419	2.83
	90th percentile latency	ms	term	3.62229	3.32
	99th percentile latency	ms	term	4.03812	3.96
	99.9th percentile latency	ms	term	5.9753	4.33
	100th percentile latency	ms	term	8.03321	5.70
	50th percentile service time	ms	term	2.49755	2.08
	90th percentile service time	ms	term	2.71322	2.53
	99th percentile service time	ms	term	3.20673	2.99
	99.9th percentile service time	ms	term	5.17998	3.37
	100th percentile service time	ms	term	6.95227	5.24

	error rate	%	term	0	0
Phrase query	Min Throughput	ops/s	phrase	110.05	110.
	Median Throughput	ops/s	phrase	110.07	110.
	Max Throughput	ops/s	phrase	110.12	110.
	50th percentile latency	ms	phrase	3.09905	2.74
	90th percentile latency	ms	phrase	3.62549	3.30
	99th percentile latency	ms	phrase	4.55457	4.81
	99.9th percentile latency	ms	phrase	8.29519	5.57
	100th percentile latency	ms	phrase	9.39771	6.54
	50th percentile service time	ms	phrase	2.38248	1.98
	90th percentile service time	ms	phrase	2.77084	2.41
	99th percentile service time	ms	phrase	3.75448	4.00
	99.9th percentile	ms	phrase	7.5974	4.70

	service time				
	100th percentile service time	ms	phrase	8.98362	5.67
	error rate	%	phrase	0	0
Aggregation query without cache	Min Throughput	ops/s	country_agg_uncached	3.6	3.6
	Median Throughput	ops/s	country_agg_uncached	3.61	3.61
	Max Throughput	ops/s	country_agg_uncached	3.61	3.61
	50th percentile latency	ms	country_agg_uncached	157.466	130.
	90th percentile latency	ms	country_agg_uncached	217.148	147.
	99th percentile latency	ms	country_agg_uncached	233.185	165.
	100th percentile latency	ms	country_agg_uncached	233.227	174.
	50th percentile service time	ms	country_agg_uncached	156.197	129.
	90th percentile service time	ms	country_agg_uncached	215.852	146.
	99th percentile	ms	country_agg_uncached	232.177	164.

	service time				
	100th percentile service time	ms	country_agg_uncached	232.321	172.
	error rate	%	country_agg_uncached	0	0
Aggregation query with cache	Min Throughput	ops/s	country_agg_cached	100.03	100.
	Median Throughput	ops/s	country_agg_cached	100.05	100.
	Max Throughput	ops/s	country_agg_cached	100.08	100.
	50th percentile latency	ms	country_agg_cached	2.44457	2.29
	90th percentile latency	ms	country_agg_cached	2.97922	3.57
	99th percentile latency	ms	country_agg_cached	3.96393	3.91
	99.9th percentile latency	ms	country_agg_cached	5.3294	4.19
	100th percentile latency	ms	country_agg_cached	7.9529	4.51
	50th percentile service time	ms	country_agg_cached	1.71924	1.57
	90th percentile service time	ms	country_agg_cached	1.97892	1.89

	99th percentile service time	ms	country_agg_cached	2.22611	2.19
	99.9th percentile service time	ms	country_agg_cached	5.0967	3.42
	100th percentile service time	ms	country_agg_cached	7.02246	4.28
	error rate	%	country_agg_cached	0	0
Paged pull	Min Throughput	pages/s	scroll	20.04	20.0
	Median Throughput	pages/s	scroll	20.04	20.0
	Max Throughput	pages/s	scroll	20.05	20.0
	50th percentile latency	ms	scroll	576.675	538.
	90th percentile latency	ms	scroll	585.156	543.
	99th percentile latency	ms	scroll	598.95	582.
	100th percentile latency	ms	scroll	602.009	584.
	50th percentile service time	ms	scroll	575.118	537.

	90th percentile service time	ms	scroll	583.906	542.
	99th percentile service time	ms	scroll	597.482	580.
	100th percentile service time	ms	scroll	600.578	583.
	error rate	%	scroll	0	0
Script query (using expression script)	Min Throughput	ops/s	expression	2	2
	Median Throughput	ops/s	expression	2	2
	Max Throughput	ops/s	expression	2	2
	50th percentile latency	ms	expression	299.685	265.
	90th percentile latency	ms	expression	416.613	287.
	99th percentile latency	ms	expression	465.776	311.
	100th percentile latency	ms	expression	468.083	391.
	50th percentile service time	ms	expression	298.594	264.
	90th	ms	expression	415.045	285.

	percentile service time				
	99th percentile service time	ms	expression	464.598	310.
	100th percentile service time	ms	expression	467.106	390.
	error rate	%	expression	0	0
Script query (using painless static script without dynamically getting field values)	Min Throughput	ops/s	painless_static	1.5	1.5
	Median Throughput	ops/s	painless_static	1.5	1.5
	Max Throughput	ops/s	painless_static	1.5	1.5
	50th percentile latency	ms	painless_static	383.485	337.
	90th percentile latency	ms	painless_static	514.495	358.
	99th percentile latency	ms	painless_static	561.342	375.
	100th percentile latency	ms	painless_static	568.066	395.
	50th percentile service time	ms	painless_static	382.158	337.
	90th percentile	ms	painless_static	513.202	357.

	service time				
	99th percentile service time	ms	painless_static	560.61	374.
	100th percentile service time	ms	painless_static	567.419	394.
	error rate	%	painless_static	0	0
Script query (using painless static script with dynamically getting field values)	Min Throughput	ops/s	painless_dynamic	1.5	1.5
	Median Throughput	ops/s	painless_dynamic	1.5	1.5
	Max Throughput	ops/s	painless_dynamic	1.5	1.5
	50th percentile latency	ms	painless_dynamic	377.278	334.
	90th percentile latency	ms	painless_dynamic	517.496	354.
	99th percentile latency	ms	painless_dynamic	576.697	377.
	100th percentile latency	ms	painless_dynamic	580.017	381.
	50th percentile service time	ms	painless_dynamic	376.339	333.
	90th percentile	ms	painless_dynamic	516.407	353.

	service time				
	99th percentile service time	ms	painless_dynamic	575.714	375.
	100th percentile service time	ms	painless_dynamic	579.642	379.
	error rate	%	painless_dynamic	0	0
Geographic range query (based on Gaussian decay function)	Min Throughput	ops/s	decay_geo_gauss_function_score	1	1
	Median Throughput	ops/s	decay_geo_gauss_function_score	1	1
	Max Throughput	ops/s	decay_geo_gauss_function_score	1	1
	50th percentile latency	ms	decay_geo_gauss_function_score	348.531	327.
	90th percentile latency	ms	decay_geo_gauss_function_score	398.351	336.
	99th percentile latency	ms	decay_geo_gauss_function_score	411.483	343.
	100th percentile latency	ms	decay_geo_gauss_function_score	457.615	344.
	50th percentile service time	ms	decay_geo_gauss_function_score	346.881	326.
	90th percentile	ms	decay_geo_gauss_function_score	397.08	336.

	service time				
	99th percentile service time	ms	decay_geo_gauss_function_score	410.421	342.
	100th percentile service time	ms	decay_geo_gauss_function_score	455.704	342.
	error rate	%	decay_geo_gauss_function_score	0	0
Geographic range query (based on Gaussian decay function with dynamically getting field values through script)	Min Throughput	ops/s	decay_geo_gauss_script_score	1	1
	Median Throughput	ops/s	decay_geo_gauss_script_score	1	1
	Max Throughput	ops/s	decay_geo_gauss_script_score	1	1
	50th percentile latency	ms	decay_geo_gauss_script_score	368.275	341.
	90th percentile latency	ms	decay_geo_gauss_script_score	414.905	349.
	99th percentile latency	ms	decay_geo_gauss_script_score	468.888	354.
	100th percentile latency	ms	decay_geo_gauss_script_score	477.25	364.
	50th percentile service time	ms	decay_geo_gauss_script_score	366.945	339.
	90th percentile	ms	decay_geo_gauss_script_score	413.609	348.

	service time				
	99th percentile service time	ms	decay_geo_gauss_script_score	467.627	353.
	100th percentile service time	ms	decay_geo_gauss_script_score	475.367	362.
	error rate	%	decay_geo_gauss_script_score	0	0
Custom scoring function query (defining function based on field value)	Min Throughput	ops/s	field_value_function_score	1.5	1.5
	Median Throughput	ops/s	field_value_function_score	1.5	1.5
	Max Throughput	ops/s	field_value_function_score	1.51	1.51
	50th percentile latency	ms	field_value_function_score	139.661	120.
	90th percentile latency	ms	field_value_function_score	183.675	137.
	99th percentile latency	ms	field_value_function_score	197.653	147.
	100th percentile latency	ms	field_value_function_score	202.345	169.
	50th percentile service time	ms	field_value_function_score	138.423	119.
	90th percentile	ms	field_value_function_score	182.404	136.

	service time				
	99th percentile service time	ms	field_value_function_score	196.734	146.
	100th percentile service time	ms	field_value_function_score	201.442	168.
	error rate	%	field_value_function_score	0	0
Custom scoring function query (dynamically getting field values through script to calculate scores)	Min Throughput	ops/s	field_value_script_score	1.5	1.5
	Median Throughput	ops/s	field_value_script_score	1.5	1.5
	Max Throughput	ops/s	field_value_script_score	1.51	1.51
	50th percentile latency	ms	field_value_script_score	188.952	168.
	90th percentile latency	ms	field_value_script_score	264.095	178.
	99th percentile latency	ms	field_value_script_score	271.153	196.
	100th percentile latency	ms	field_value_script_score	271.901	198.
	50th percentile service time	ms	field_value_script_score	187.218	166.
	90th percentile	ms	field_value_script_score	263.207	177.

	service time				
	99th percentile service time	ms	field_value_script_score	269.578	195.
	100th percentile service time	ms	field_value_script_score	270.138	197.
	error rate	%	field_value_script_score	0	0
Large terms query	Min Throughput	ops/s	large_terms	1.1	1.1
	Median Throughput	ops/s	large_terms	1.1	1.1
	Max Throughput	ops/s	large_terms	1.1	1.1
	50th percentile latency	ms	large_terms	265.007	241.
	90th percentile latency	ms	large_terms	296.009	252.
	99th percentile latency	ms	large_terms	310.358	265.
	100th percentile latency	ms	large_terms	311.049	272.
	50th percentile service time	ms	large_terms	256.372	233.
	90th percentile	ms	large_terms	287.851	244.

	service time				
	99th percentile service time	ms	large_terms	301.827	258.
	100th percentile service time	ms	large_terms	302.251	264.
	error rate	%	large_terms	0	0
Large filtered terms query	Min Throughput	ops/s	large_filtered_terms	1.1	1.1
	Median Throughput	ops/s	large_filtered_terms	1.1	1.1
	Max Throughput	ops/s	large_filtered_terms	1.1	1.1
	50th percentile latency	ms	large_filtered_terms	268.135	233.
	90th percentile latency	ms	large_filtered_terms	304.158	241.
	99th percentile latency	ms	large_filtered_terms	351.209	251.
	100th percentile latency	ms	large_filtered_terms	352.003	260.
	50th percentile service time	ms	large_filtered_terms	259.546	225.
	90th percentile	ms	large_filtered_terms	295.721	233.

	service time				
	99th percentile service time	ms	large_filtered_terms	342.342	243.
	100th percentile service time	ms	large_filtered_terms	343.378	252.
	error rate	%	large_filtered_terms	0	0
Large prohibited terms query	Min Throughput	ops/s	large_prohibited_terms	1.1	1.1
	Median Throughput	ops/s	large_prohibited_terms	1.1	1.1
	Max Throughput	ops/s	large_prohibited_terms	1.1	1.1
	50th percentile latency	ms	large_prohibited_terms	270.041	235.
	90th percentile latency	ms	large_prohibited_terms	310.351	241.
	99th percentile latency	ms	large_prohibited_terms	347.414	255.
	100th percentile latency	ms	large_prohibited_terms	349.499	259.
	50th percentile service time	ms	large_prohibited_terms	261.734	227.
	90th percentile	ms	large_prohibited_terms	302.279	233.

	service time				
	99th percentile service time	ms	large_prohibited_terms	339.278	248.
	100th percentile service time	ms	large_prohibited_terms	340.817	251.
	error rate	%	large_prohibited_terms	0	0
Descending order query	Min Throughput	ops/s	desc_sort_population	1.5	1.5
	Median Throughput	ops/s	desc_sort_population	1.51	1.51
	Max Throughput	ops/s	desc_sort_population	1.51	1.51
	50th percentile latency	ms	desc_sort_population	58.5828	48.3
	90th percentile latency	ms	desc_sort_population	77.9981	63.0
	99th percentile latency	ms	desc_sort_population	80.8863	71.7
	100th percentile latency	ms	desc_sort_population	83.1661	83.3
	50th percentile service time	ms	desc_sort_population	57.1212	47.0
	90th percentile	ms	desc_sort_population	76.7082	61.3

	service time				
	99th percentile service time	ms	desc_sort_population	79.2907	70.4
	100th percentile service time	ms	desc_sort_population	81.6364	81.6
	error rate	%	desc_sort_population	0	0
Ascending order query	Min Throughput	ops/s	asc_sort_population	1.5	1.5
	Median Throughput	ops/s	asc_sort_population	1.51	1.51
	Max Throughput	ops/s	asc_sort_population	1.51	1.51
	50th percentile latency	ms	asc_sort_population	62.4328	49.2
	90th percentile latency	ms	asc_sort_population	79.8441	67.5
	99th percentile latency	ms	asc_sort_population	83.9411	84.6
	100th percentile latency	ms	asc_sort_population	84.3925	85.8
	50th percentile service time	ms	asc_sort_population	61.0637	47.8
	90th percentile	ms	asc_sort_population	78.4101	66.0

	service time				
	99th percentile service time	ms	asc_sort_population	82.2652	83.6
	100th percentile service time	ms	asc_sort_population	82.5616	84.2
	error rate	%	asc_sort_population	0	0
search_after query with sorting in ascending order	Min Throughput	ops/s	asc_sort_with_after_population	1.5	1.5
	Median Throughput	ops/s	asc_sort_with_after_population	1.51	1.51
	Max Throughput	ops/s	asc_sort_with_after_population	1.51	1.51
	50th percentile latency	ms	asc_sort_with_after_population	88.1871	99.1
	90th percentile latency	ms	asc_sort_with_after_population	127.995	86.0
	99th percentile latency	ms	asc_sort_with_after_population	131.171	102.
	100th percentile latency	ms	asc_sort_with_after_population	132.181	106.
	50th percentile service time	ms	asc_sort_with_after_population	87.132	68.2
	90th percentile	ms	asc_sort_with_after_population	126.818	84.6

	service time				
	99th percentile service time	ms	asc_sort_with_after_population	129.453	101.
	100th percentile service time	ms	asc_sort_with_after_population	130.452	105.
	error rate	%	asc_sort_with_after_population	0	0
Query with sorting high base fields in descending order (quickly getting topK based on DistanceFeatureQuery)	Min Throughput	ops/s	desc_sort_geonameid	6.02	6.02
	Median Throughput	ops/s	desc_sort_geonameid	6.02	6.02
	Max Throughput	ops/s	desc_sort_geonameid	6.03	6.03
	50th percentile latency	ms	desc_sort_geonameid	7.4659	5.53
	90th percentile latency	ms	desc_sort_geonameid	8.26766	6.20
	99th percentile latency	ms	desc_sort_geonameid	8.72369	6.67
	100th percentile latency	ms	desc_sort_geonameid	8.79956	6.95
	50th percentile service time	ms	desc_sort_geonameid	6.59986	4.61
	90th percentile	ms	desc_sort_geonameid	7.24539	5.45

	service time				
	99th percentile service time	ms	desc_sort_geonameid	7.57925	5.65
	100th percentile service time	ms	desc_sort_geonameid	7.64471	5.65
	error rate	%	desc_sort_geonameid	0	0
search_after query with sorting high base fields in descending order	Min Throughput	ops/s	desc_sort_with_after_geonameid	6.01	6.01
	Median Throughput	ops/s	desc_sort_with_after_geonameid	6.01	6.02
	Max Throughput	ops/s	desc_sort_with_after_geonameid	6.02	6.02
	50th percentile latency	ms	desc_sort_with_after_geonameid	89.4587	56.5
	90th percentile latency	ms	desc_sort_with_after_geonameid	119.777	79.6
	99th percentile latency	ms	desc_sort_with_after_geonameid	123.271	87.7
	100th percentile latency	ms	desc_sort_with_after_geonameid	123.628	89.3
	50th percentile service time	ms	desc_sort_with_after_geonameid	88.512	55.4
	90th percentile	ms	desc_sort_with_after_geonameid	118.72	79.2

	service time				
	99th percentile service time	ms	desc_sort_with_after_geonameid	122.79	87.3
	100th percentile service time	ms	desc_sort_with_after_geonameid	122.791	88.3
	error rate	%	desc_sort_with_after_geonameid	0	0
Query with sorting high base fields in ascending order (quickly getting topK based on DistanceFeatureQuery)	Min Throughput	ops/s	asc_sort_geonameid	6.02	6.02
	Median Throughput	ops/s	asc_sort_geonameid	6.02	6.02
	Max Throughput	ops/s	asc_sort_geonameid	6.03	6.03
	50th percentile latency	ms	asc_sort_geonameid	5.80593	5.19
	90th percentile latency	ms	asc_sort_geonameid	6.55438	5.74
	99th percentile latency	ms	asc_sort_geonameid	7.36432	6.22
	100th percentile latency	ms	asc_sort_geonameid	7.49672	11.6
	50th percentile service time	ms	asc_sort_geonameid	4.91916	4.35
	90th percentile	ms	asc_sort_geonameid	5.61126	4.92

	service time				
	99th percentile service time	ms	asc_sort_geonameid	6.12285	5.38
	100th percentile service time	ms	asc_sort_geonameid	6.51222	10.6
	error rate	%	asc_sort_geonameid	0	0
search_after query with sorting high base fields in ascending order	Min Throughput	ops/s	asc_sort_with_after_geonameid	6.01	6.01
	Median Throughput	ops/s	asc_sort_with_after_geonameid	6.01	6.02
	Max Throughput	ops/s	asc_sort_with_after_geonameid	6.01	6.02
	50th percentile latency	ms	asc_sort_with_after_geonameid	70.994	58.1
	90th percentile latency	ms	asc_sort_with_after_geonameid	104.817	76.5
	99th percentile latency	ms	asc_sort_with_after_geonameid	108.797	91.6
	100th percentile latency	ms	asc_sort_with_after_geonameid	108.929	91.6
	50th percentile service time	ms	asc_sort_with_after_geonameid	69.7056	57.1
	90th percentile	ms	asc_sort_with_after_geonameid	103.875	75.7

	service time				
	99th percentile service time	ms	asc_sort_with_after_geonameid	107.828	91.1
	100th percentile service time	ms	asc_sort_with_after_geonameid	108.539	91.3
	error rate	%	asc_sort_with_after_geonameid	0	0

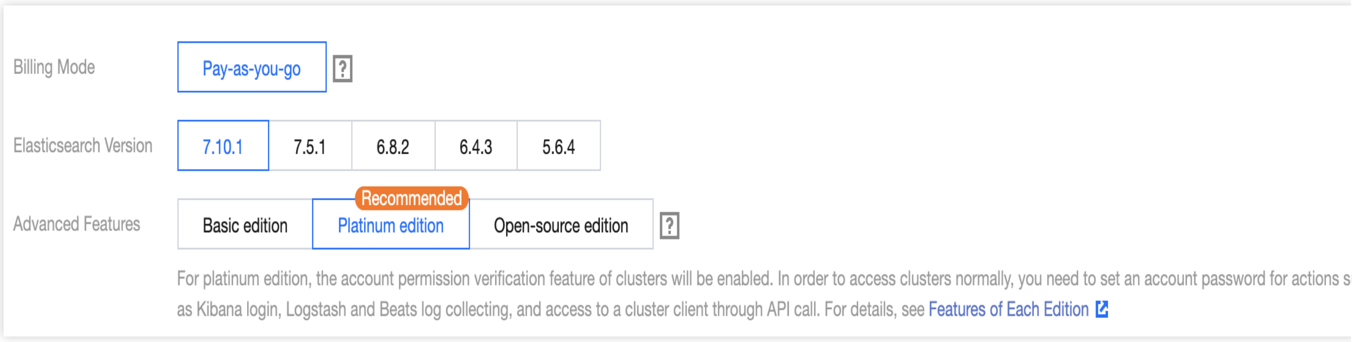
Elastic Stack (X-Pack)

Last updated：2021-10-29 15:04:01

Overview

X-Pack features are Elasticsearch's official commercial features, including security, SQL, machine learning, and monitoring. It facilitates the application development and OPS management of Elasticsearch services. ES offers editions that come with such features, which you can select when purchasing and creating a cluster. The features in different editions are detailed below.

Purchase Guide


Billing Mode: Pay-as-you-go
Elasticsearch Version: 7.10.1 | 7.5.1 | 6.8.2 | 6.4.3 | 5.6.4
Advanced Features: Basic edition | Recommended Platinum edition | Open-source edition

For platinum edition, the account permission verification feature of clusters will be enabled. In order to access clusters normally, you need to set an account password for actions s as Kibana login, Logstash and Beats log collecting, and access to a cluster client through API call. For details, see Features of Each Edition

As shown in the figure above, there are options for the X-Pack features on the ES purchase page. ES offers three editions that have different X-Pack features as follows:

Item	Basic	Platinum	Open Source
X-Pack included	✓	✓	×
X-Pack completeness	Partial	All	None

Purchase recommendation In order to be able to use more advanced features in ES, we recommend that you choose the **Platinum Edition** when you create a cluster. The specific features and differences of each edition are detailed below. For pricing information, please see Product Pricing.

X-Pack Overview

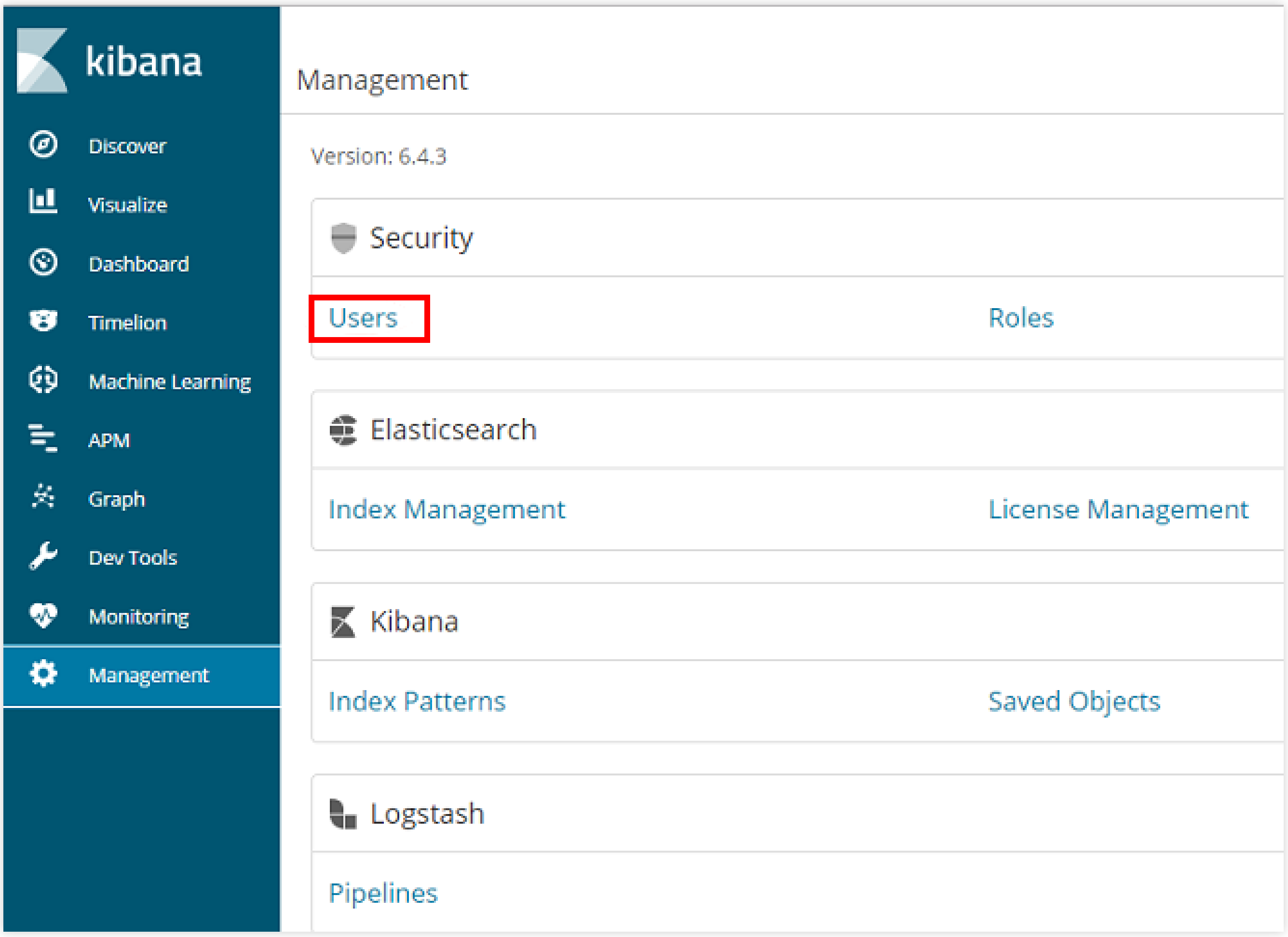
This document describes some of the commonly used X-Pack features. For more information, please see Elasticsearch's official Elastic Stack subscriptions and API documentation.

Note:

Some features vary by editions (Basic, Platinum, and Open Source).

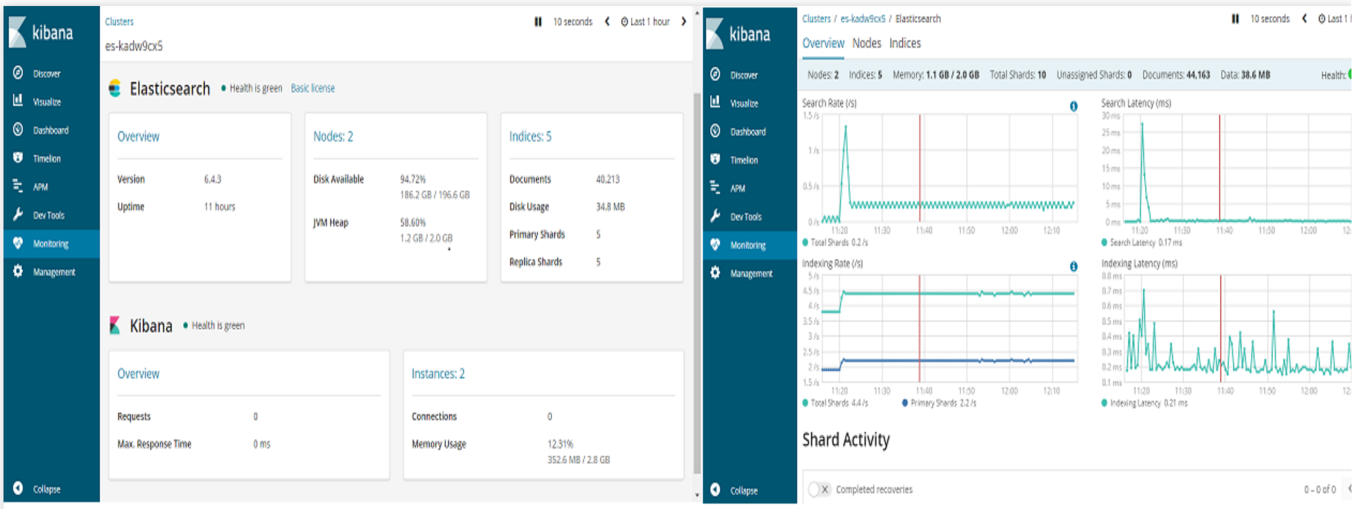
Some features are unavailable in earlier ES versions. For more information, please submit a ticket.

Security This feature supports refined read/write permission control at the index and field levels and effectively protects data security by enabling data security protection and business access isolation, granting access to the right people, and preventing malicious attacks and data leakage.

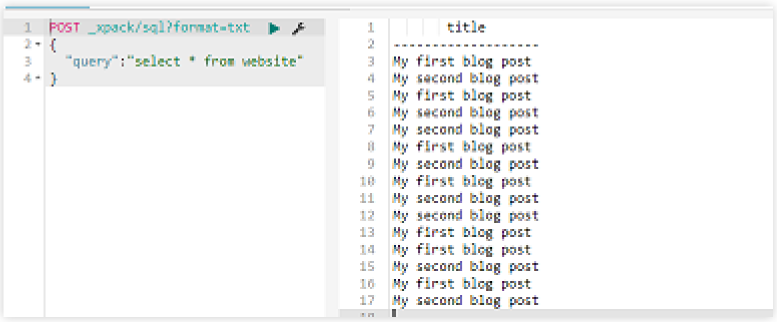


Machine learning In the application scenario of custom data alerting, it is sometimes difficult to set rules and thresholds to define the changes. In this case, the trend in data changes and reasonable fluctuation range can be predicted by the unattended machine learning feature, and when the data deviates from the normal trend, alarms will be triggered and notifications sent.

Monitoring Monitoring information can be comprehensively collected at multiple levels such as cluster, node, and index, helping you understand the cluster operations in real time and facilitating your application development and OPS.



SQL This feature makes full-text search and statistical analysis of Elasticsearch data possible through traditional database SQL tools. CLI and REST access methods are supported. **The Platinum Edition further supports JDBC connection**. This feature enables you to seamlessly connect ES with your existing business systems and thus reduces your learning costs for new technologies.



```
POST _xpack/sql?format=txt
{
  "query":"select * from website"
}
```

```
      title
-------------------
My first blog post
My second blog post
My first blog post
My second blog post
My second blog post
My first blog post
My second blog post
My first blog post
My second blog post
My second blog post
My first blog post
My first blog post
My second blog post
My first blog post
My second blog post
```

Note:

In terms of SQL support, the Open Source Edition integrates with other SQL plugins. For more information, please see elasticsearch-sql.

Detailed comparison among editions

This section mainly compares and highlights some key features of different Elasticsearch versions. As Elasticsearch is in a stage of rapid development, and the support for various features by different versions is constantly adjusted, we do not guarantee that the following content can stay in sync with the changes in the community.

For the latest and most accurate feature comparison, please see Elasticsearch's official Elastic Stack subscriptions.

Note:

In the table below,

●

,

◑ , and

— are used to indicate the feature completeness.

● : all;

◑ : partial;

— : none.

Module	Feature	Open Source	Basic	Platinum
Elasticsearch	Scalability and resiliency	◑	◑	●
	Query and analytics	◑	◑	●
	Data enrichment	●	●	●
	Management and tooling	◑	◑	●
	Security	—	—	●
	Machine Learning			

		—	—	●
Kibana	Explore and visualize	◐	◐	●
	Stack management and tooling	◐	◐	●
	Stack monitoring	—	◐	●
	Share and collaborate	◐	◐	●
	Security	—	—	●
	Machine learning	—	—	●
Beats	Data collection	◐	◐	●
	Data shipping	◐	◐	●
	Module			

		◑	◑	●
	Monitoring and management	—	◑	●
Logstash	Data collection	●	●	●
	Data enrichment	●	●	●
	Data shipping	●	●	●
	Module	◑	●	●
	Monitoring and management	—	◑	●
ELASTIC APM	APM server	●	●	●
	APM agents	●	●	●
	APM dashboards in Kibana			

		●	●	●
	APM UI	—	●	●
	Distributed tracing	—	●	●
	Machine learning integration	—	—	●
Elastic Logs	Log shipper (Filebeat)	●	●	●
	Dashboards for common data sources	●	●	●
	Logs UI	—	●	●
Elastic Infrastructure	Metric shipper (Metricbeat)	●	●	●
	Dashboards for common data sources	●	●	●
	Infrastructure UI			

		—	●	●
Elastic Uptime	Uptime monitor (Heartbeat)	●	●	●
	Uptime dashboards in Kibana	●	●	●
	Uptime UI	—	●	●

Detailed descriptions of certain Elasticsearch features:

Note:

In the table below, ✓ means the feature is available, - means not available.

Elasticsearch Feature Module	Item	Open Source	Basic	Platinum
Management and Tooling	REST APIs	✓	✓	✓
	Language clients	✓	✓	✓
	Snapshot/restore	✓	✓	✓
	_source only snapshot	-	✓	✓
	SQL interpreter CLI	-	✓	✓
	Data rollups	-	✓	✓
	Index lifecycle management	-	✓	✓
	Frozen indices	-	✓	✓
	Upgrade Assistant APIs	-	✓	✓
	JDBC client	-	-	✓
	ODBC client	-	-	✓
Security	Encrypted communications	-	✓	✓

	Role-based access control	-	✓	✓
	File and native authentication	-	✓	✓
	Audit logging	-	-	✓
	Attribute-based access control	-	-	✓
	Field- and document-level security	-	-	✓
Machine Learning	Anomaly detection on time series	-	-	✓
	Population/entity analysis	-	-	✓
	Log message categorization	-	-	✓
	Root cause indication	-	-	✓
	Alerting on anomalies	-	-	✓
	Forecasting on time series	-	-	✓

Strengths

Last updated：2020-08-03 11:24:48

ES is hosted in the cloud, making it easy for you to create and manage Elasticsearch clusters and ensure high availability in production environments. Its core benefits are detailed below:

Ease of Deployment and Management

An ES cluster can be created in a few minutes without deploying software and hardware. Additionally, ES comes with a cluster management tool called Kibana, which assist cluster and alert systems to facilitate daily operation and management.

Elastic Scaling

ES has various node specifications and storage media for different business needs. Clusters can be scaled up or down to meet your current business needs and control costs.

Elasticsearch X-Pack

ES integrates Elasticsearch X-Pack, which has advanced features such as security, SQL, and machine learning to improve the efficiency of security management, usage, and OPS of Elasticsearch clusters.

High Availability

ES can be deployed in multiple availability zones, guaranteeing service continuity in the event of force majeure such as network or power failure in one single availability zone. A COS data backup policy can periodically back up the data to ensure rapid restoration in case of data loss due to unexpected conditions. In addition, ES boasts specially created policies such as kernel optimization that help comprehensively ensure data security and service stability.

Security Reinforcement

ES can be deployed in a logically isolated VPC, giving you full control over your environment configuration and the ability to customize network access control lists and security groups. It features a blocklisting/allowlisting mechanism for Kibana and IP access requests, and the security capability of X-Pack enables access control at the field level, helping ensure the security of your resources in the cloud.

Openness and Service Integration

ES supports the complete system of ELK products and is compatible with standard open-source RESTful APIs and ecosystem components. It can be integrated with Tencent Cloud products such as COS, FL, CMQ, and TencentDB to implement data transfer and backup to meet your needs in different business scenarios.

Scenarios

Last updated：2024-11-29 19:33:02

Log Analysis

Devices such as website servers, mobile devices, and IOT sensors can generate a high number of logs in various types that are stored on scattered nodes, which poses a great challenge to services relying on log search like troubleshooting and business analysis. ES provides a flexible, scalable, real-time, and centralized storage scheme and a full-text search feature to facilitate unified management and query of logs, helping you quickly identify and locate problems and improve the troubleshooting efficiency.

Full-Text Search

In-site search service for massive amounts of data such as search for ecommerce items, mobile apps, and organizational information is a necessary way to get the desired information efficiently. ES has a full-text search feature that can retrieve structured and unstructured data with ease. It also provides simple and convenient RESTful APIs and clients in various programming languages to help you quickly build a stable search service and integrate it into your existing business framework.

Business Intelligence (BI)

While data-driven operation is gaining popularity in today's industry landscape, businesses such as ecommerce, mobile app, advertising, and media need to rely on data analysis and data mining as assistance in business decision-making; however, large-scale business data poses great challenges to statistics collection and analysis. To cope with this problem, ES provides a structured query feature and supports complex filtering and aggregated statistics, which helps you efficiently perform statistical analysis on massive amounts of data, discover problems and opportunities, assist in business decision-making, and tap into the value of data.

Capabilities and Restrictions

Last updated：2024-11-29 19:34:39

ES is a cloud-based PaaS service developed based on open-source Elasticsearch. It enables you to quickly build an Elasticsearch cluster service to develop applications such as log analysis and data search. The following describes its capabilities and use limits.

Product Composition

ES consists of ES cluster, its core component, and Kibana, its visual data analysis tool. For data collection and transfer to the ES cluster, you can deploy data collection tools such as Beats and Logstash or develop custom applications based on your own business needs.

Available Configurations

Node specification

Parameter Name	CPU Cores	Memory
ES.S1.SMALL2	1	2 GB
ES.S1.MEDIUM4	2	4 GB
ES.S1.MEDIUM8	2	8 GB
ES.S1.LARGE16	4	16 GB
ES.S1.2XLARGE32	8	32 GB
ES.S1.4XLARGE32	16	32 GB
ES.S1.4XLARGE64	16	64 GB

The node with 1 core and 2 GB of memory is intended for testing purpose only and is not recommended for production environments.

Storage

SSD cloud disks are used. The disk capacity is 100 GB–6 TB on a single node.

Number of nodes

The number of nodes is limited to 2–50. As an ES cluster typically consists of nodes deployed in a distributed manner, a master node is required to manage it. In order to prevent the risk of split brain caused by possible node failures, you are recommended to select at least three nodes for a cluster.

Configuration selection

See Evaluation of Cluster Specification and Capacity Configuration.

Network Access

Private network access in VPC

In order to ensure data security, ES is built in your VPC, and you can only access an ES cluster from your VPC to write and query data. If you need to access a cluster over the public network for development and debugging purposes, you can connect your local IDC to the VPC using VPN Connections. In this case, please take effective measures to protect your data.

Kibana page

You can access the Kibana page over the public network. For the sake of data security, a password and access IP blocklist/allowlist need to be set for the Kibana page.

VPC network selection

Once an ES cluster is created, its VPC network cannot be changed; therefore, please make a good plan for your business deployment in advance when creating a cluster.

Related Concepts

Last updated：2020-02-26 22:15:13

An Elasticsearch cluster is generally a distributed one consisting of multiple nodes. The nodes communicate and cooperate with one another to provide searching and indexing services (client requests can be forwarded to the optimal node among all the nodes). Different nodes play one or more different roles. There are many node roles in Elasticsearch, and the most important two ones are data nodes and master nodes.

Data Node

It is mainly responsible for operations related to the storing, processing, and manipulating of data and index shards, such as I/O-, memory-, and CPU-intensive operations like CRUD, search, and aggregation. During the use of cluster, you should closely monitor the resource utilization of the data nodes and ensure cluster stability by adding more nodes to scale the cluster up when the service is overloaded.

Master Node

It is responsible for making cluster-wide operations lightweight, such as creating or deleting indices, tracking which nodes are part of which clusters, and deciding which shards to assign to which nodes. It is important to have a stable master node for the cluster health.

Master-eligible Node

This refers to a node that is eligible to be selected as a master node. Any node that meets the requirements for a master node (all nodes by default) can be selected as a master node through the master selection process.
By default, all nodes are data nodes and eligible to be a master node, which is very convenient for small clusters. Because the requests for index processing and data searching are I/O-, memory-, and CPU-intensive for data nodes, they may cause pressure on the node resources. As the cluster grows, in order to ensure that the master nodes are stable and free from pressure and to ensure the cluster stability, the master nodes should be separated from the data nodes.

Dedicated Master Node

This is a node set to serve only as a master node in an Elasticsearch cluster.

Suggestions on Dedicated Master Nodes

Configuring dedicated master nodes is mainly to ensure the stability of the cluster as it grows. It is recommended to configure at least 3 dedicated master nodes:
If the number of dedicated master nodes is 1, there is only one eligible master node.
discovery.zen.minimum_master_nodes can only be set to 1, and there is no backup in case of network failure.

If the number of dedicated master nodes is 2, there are 2 eligible master nodes. If minimum_master_nodes is set to 1, although there is a backup node, there may be a risk of split-brain (i.e., each eligible master node sets itself as the master node) when the master node is re-selected in case of network failure. If minimum_master_nodes is set to 2, as the number of eligible master nodes falls short, no master node can be selected in case of failure.
If the number of dedicated master nodes is 3, there are 3 eligible master nodes. If discovery.zen.minimum_master_nodes is set to 2, even if one eligible master node is lost in case of network failure, there is still one master node that can be re-selected.
For more information, see ES Node Description.