Java KD+-Tree Benchmark Report

Table of Contents

Performance Comparation Between Java KD+-Tree And C++ KD+-Tree

Big Data Benchmark in Windows System

Big Data Benchmark in Linux System

.

Performance Comparation Between Java KD+-Tree And C++ KD+-Tree

Performance Experiment Description

Hardware:

Memory: 64 GiB

Processor: Intel® Xeon(R) CPU E5-2630 v4 @ 2.20GHz × 40

OS Type: 64-bit

System: Linux (Fedora)

Performance Test Dataset:

Yellow 2015 Taxi data: 12 Million Records, 1.84 GB

Performance Program:

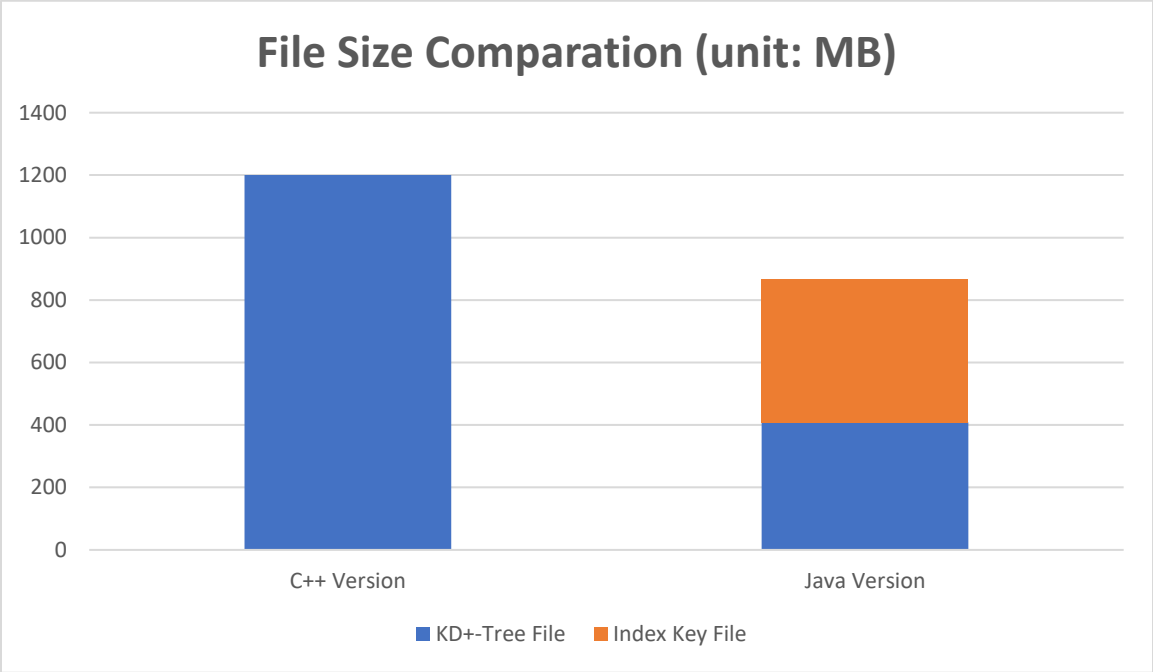
Java version KD+-Tree

C++ version KD+-Tree

Performance Comparation Tasks:

- Index files size comparation between C++ version KD+-Tree vs Java version KD+-Tree
- Building Index files time between C++ version KD+-Tree vs Java version KD+-Tree
- Taxi data range search cost time

File Size Comparation (unit: MB)

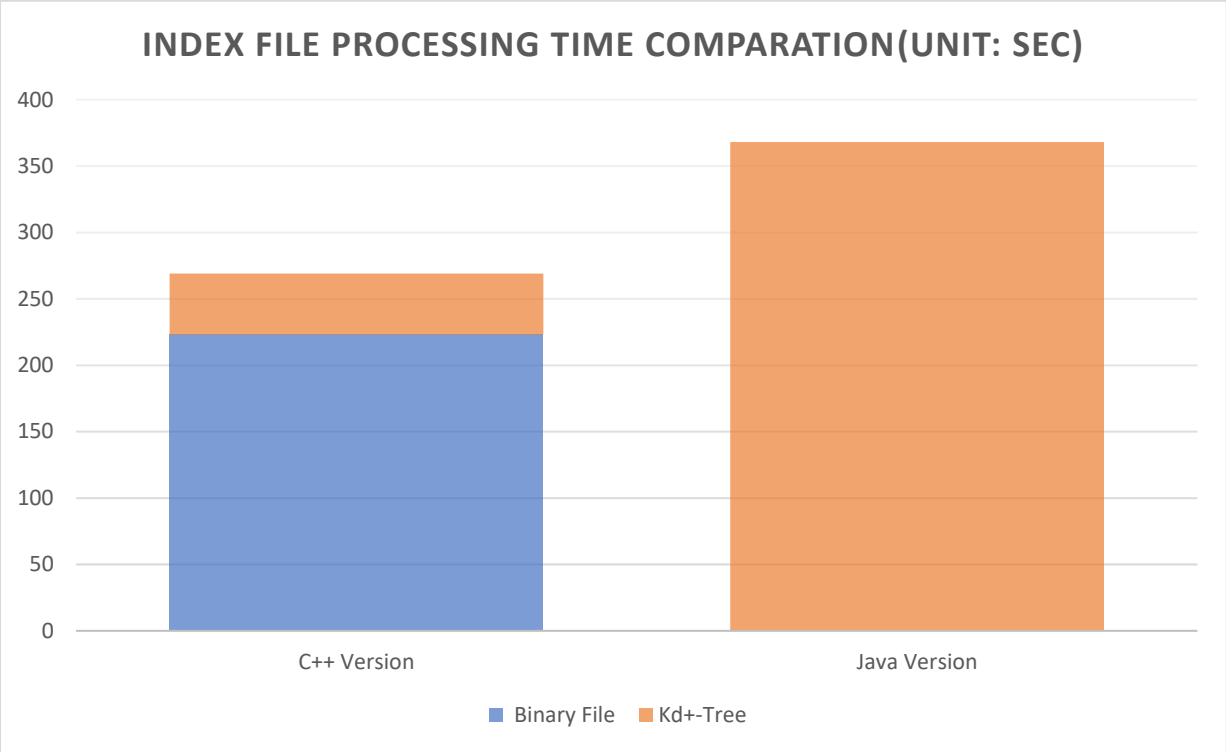


	KD+-Tree File	Index Key File
C++ Version	1200	0
Java Version	408	459

Knode size is 1 for both C++ version and Java version

C++ version's KD+-Tree contains all index key file.

Index File Processing Time Comparation (unit: SEC)

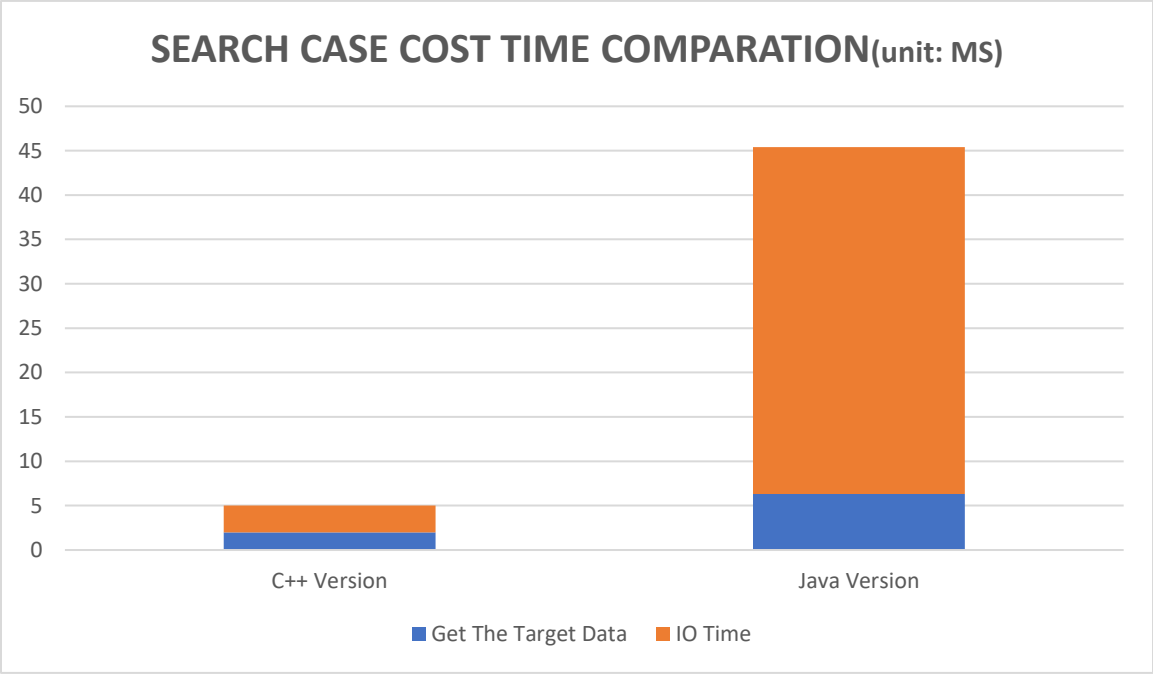


	Binary File	KD+-Tree
C++ Version	223	46
Java Version	0	475

C++ version need to extract all the data and put them to binary file with one program first. Then it can build KD+-Tree with another program. For Java version, only run one program to build the KD+-Tree index files.

Search Case Cost Time Comparation (unit: MS)

Case: 12 million records find 770 records based on the range condition



	Get the target Data	IO Time
C++ Version	2	3
Java Version	6.3	39.1

(Notice: C++ finds 758 target records since C++ version can regard float value as unsigned int value to compare. To be compared, Java version find 770 target records since it makes float become to int by multiplying 10000 which will reduce precision of original float value.)

Big Data Benchmark in Windows System

Benchmark Experiment Description

Hardware:

Computer Type: Laptop

Memory: 16 GiB

Processor: Intel® Core™ i7-7500U Processor 2.70GHz

OS Type: 64-bit

System: Windows 10 Home

Disk: SSD

Performance Test Dataset:

PUDG Game Data: 65 Million Records, 9.00 GB

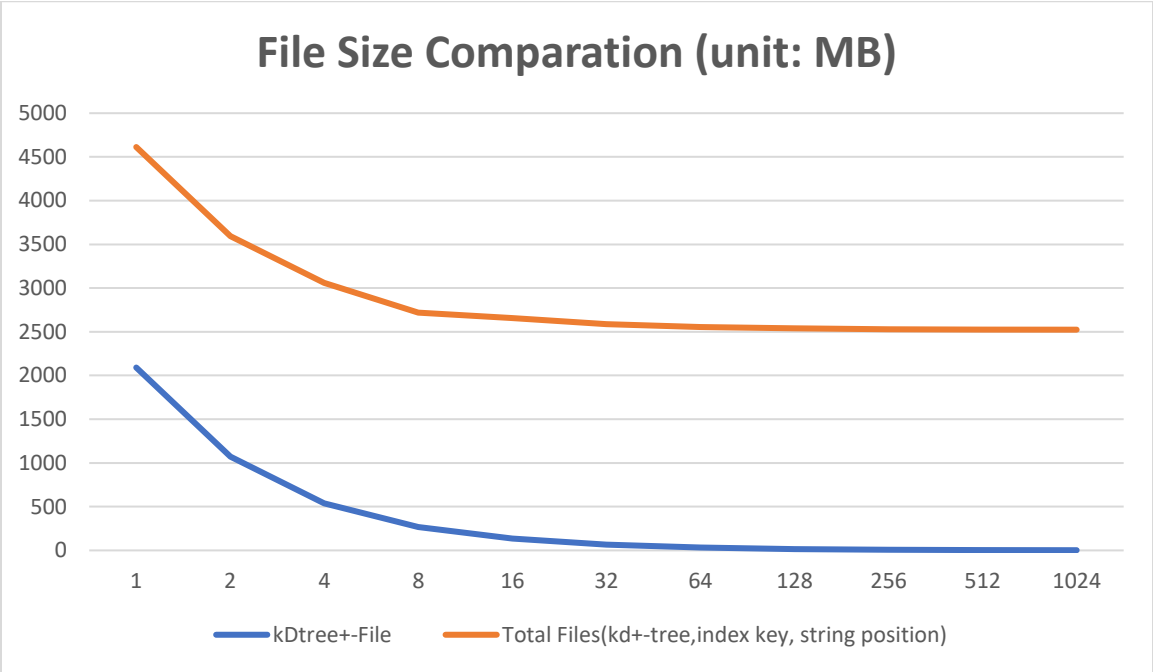
Performance Program:

Java version KD+-Tree

Performance Comparation Tasks:

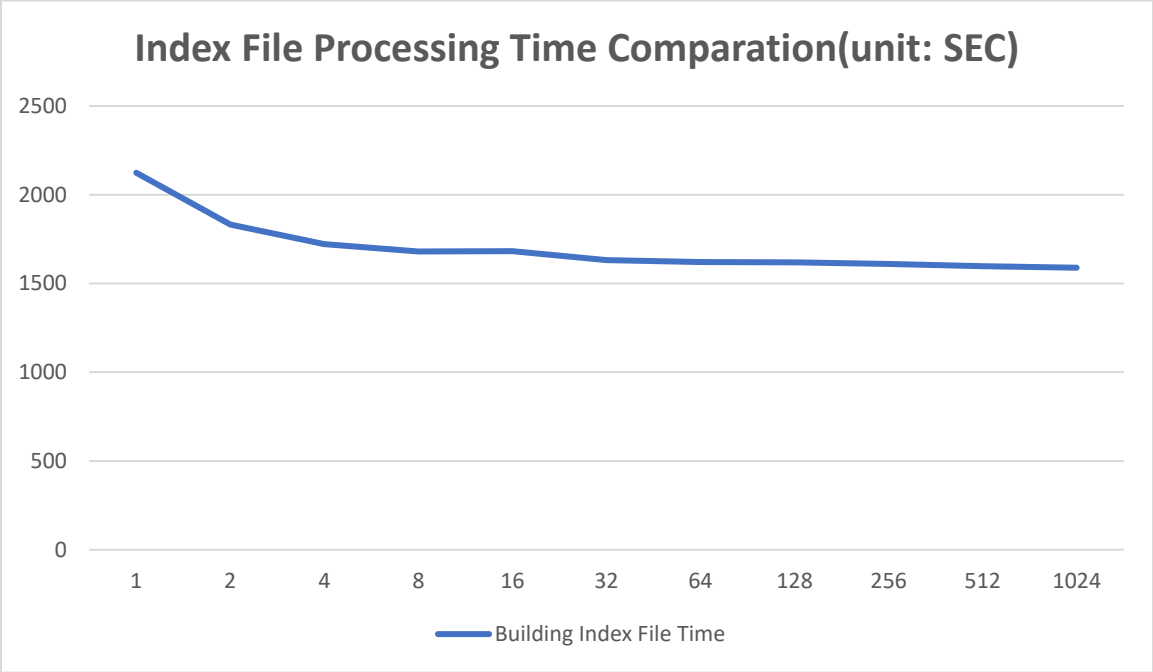
- Index files size benchmark as the knode size is different
- Building Index files time benchmark as the knode size is different
- PUDG data range search cost time
- PUDG data topK nearest POI search cost time as K = 1,10,100,500,1000,5000

File Size Comparation (unit: MB)



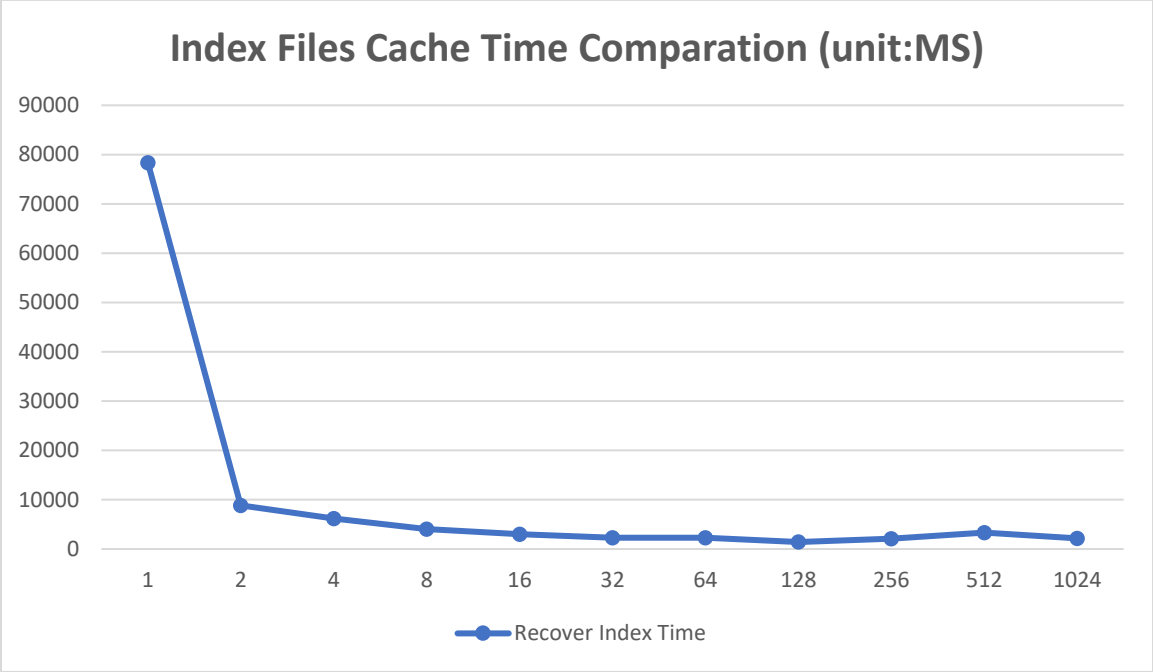
	kDtree+-File	Total Files(kD+-tree,index key, string position)
1	2091	4613
2	1073	3595
4	537	3059
8	268	2720
16	134	2656
32	67	2589
64	33	2555
128	16	2538
256	8.4	2530.4
512	4.2	2526.2
1024	2.1	2524.1

Index File Processing Time Comparation (unit: SEC)



	Building Index File Time
1	2124
2	1832
4	1722
8	1680
16	1683
32	1631
64	1621
128	1619
256	1610
512	1597
1024	1589

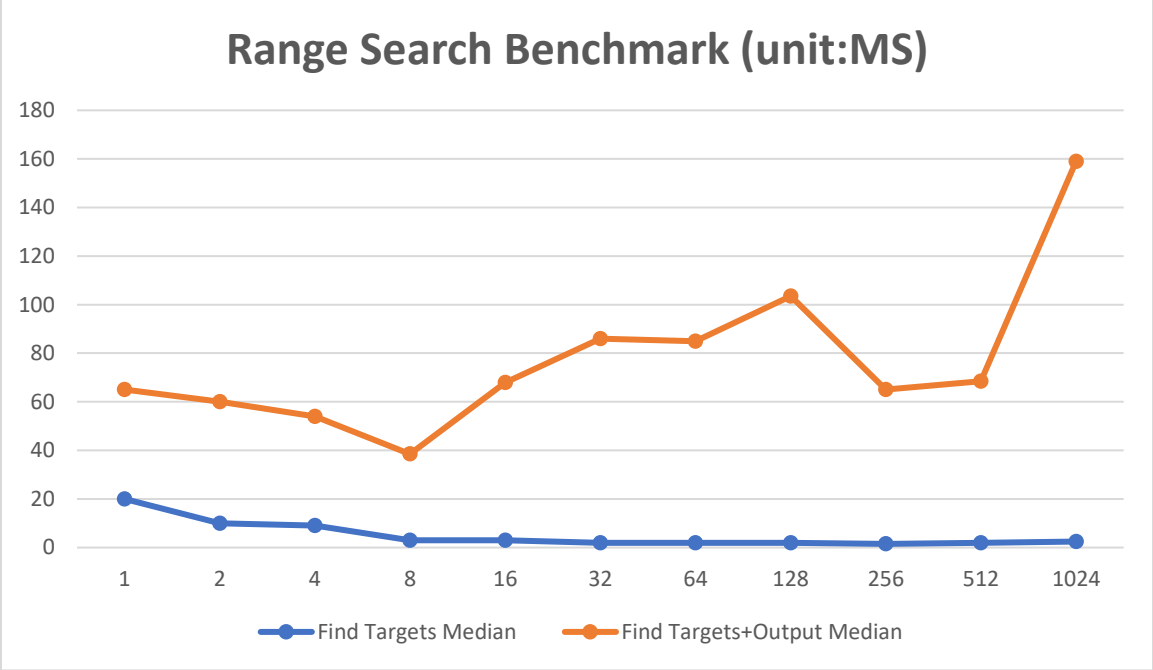
Index Files Cache Time Comparation (unit:MS)



	Recover Index Time
1	78367
2	8848
4	6172
8	4024
16	3014
32	2313
64	2314
128	1420
256	2119
512	3325
1024	2167

Range Search Benchmark (unit:MS)

Case: 65 million records find 1792 records based on the range condition



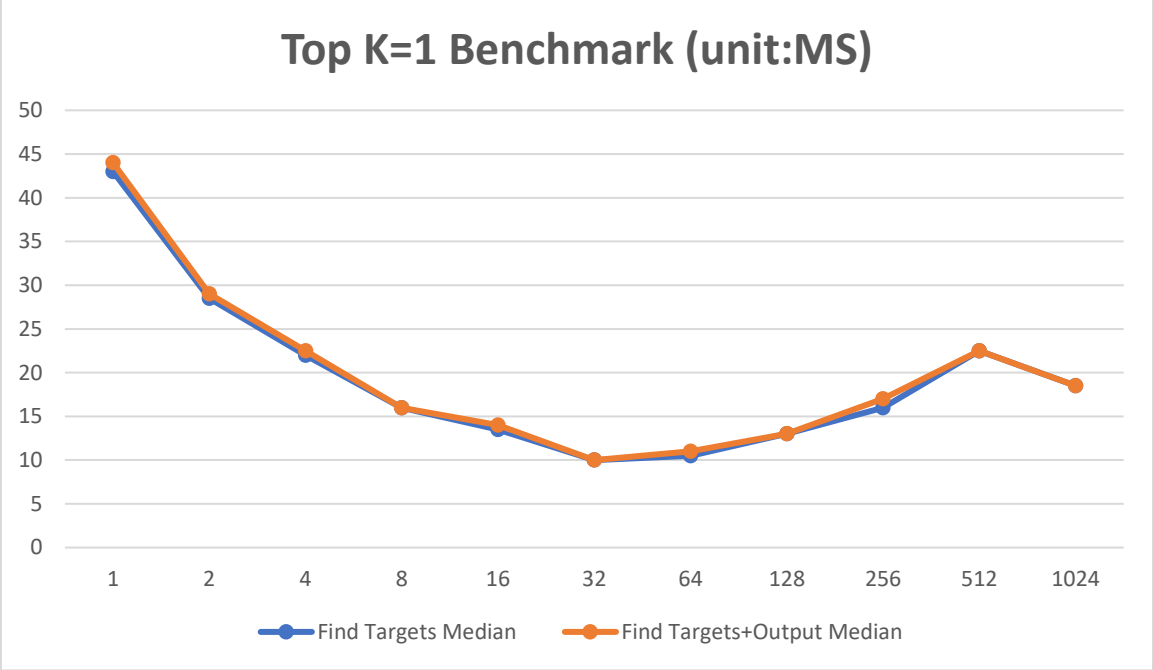
	1	2	4	8	16	32	64	128	256	512	1024
Find MIN	15	8	6	3	2	1	1	1	1	1	2
Total MIN	57	46	46	33	53	69	52	62	56	52	87
Find Median	20	10	9	3	3	2	2	2	1.5	2	2.5
Total Median	65	60	54	38.5	68	86	85	103.5	65	68.5	159
Find MAX	31	157	887	212	507	119	1311	291	396	108	49
Total MAX	15629	532	2729	1097	1657	561	4208	1517	897	445	388

Find-Find Targets

Total-Find Targets + Output

Top K=1 Benchmark (unit:MS)

Case: 65 million records find 1 record based on the top K condition



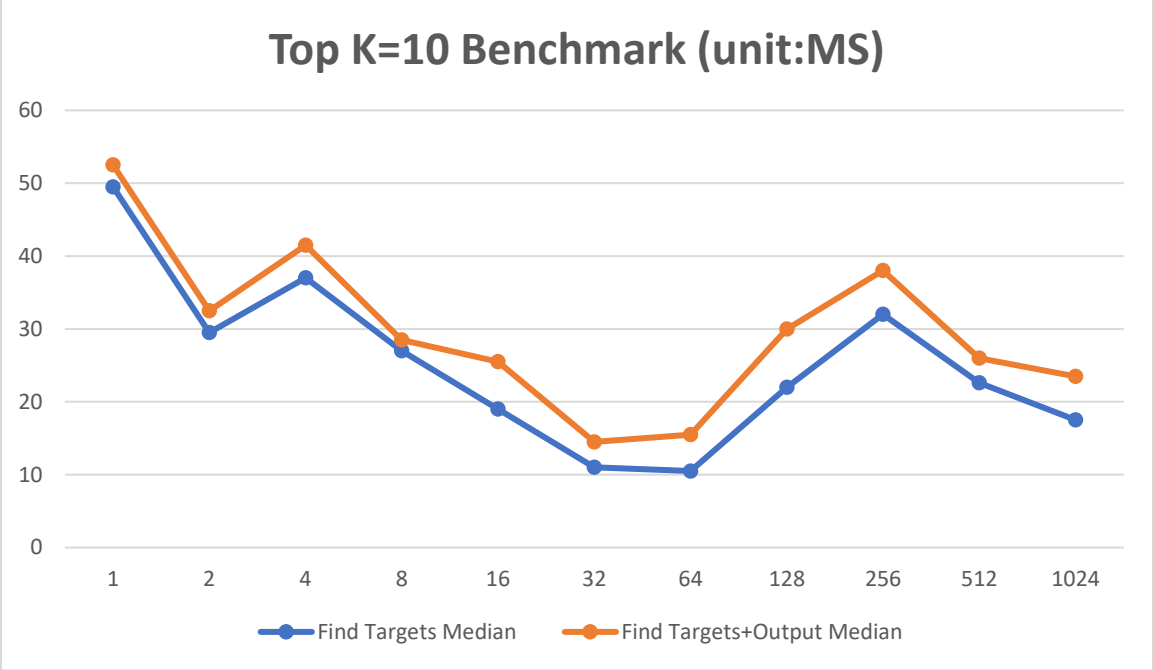
	1	2	4	8	16	32	64	128	256	512	1024
Find MIN	31	26	19	15	10	6	8	9	12	17	16
Total MIN	32	27	20	15	10	6	8	10	12	17	17
Find Median	43	28.5	22	16	13.5	10	10.5	13	16	22.5	18.5
Total Median	44	29	22.5	16	14	10	11	13	17	22.5	18.5
Find MAX	263	69	119	60	68	73	65	119	2228	120	1294
Total MAX	280	71	122	62	70	75	67	121	2407	123	1298

Find-Find Targets

Total-Find Targets + Output

Top K=10 Benchmark (unit:MS)

Case: 65 million records find 10 records based on the top K condition



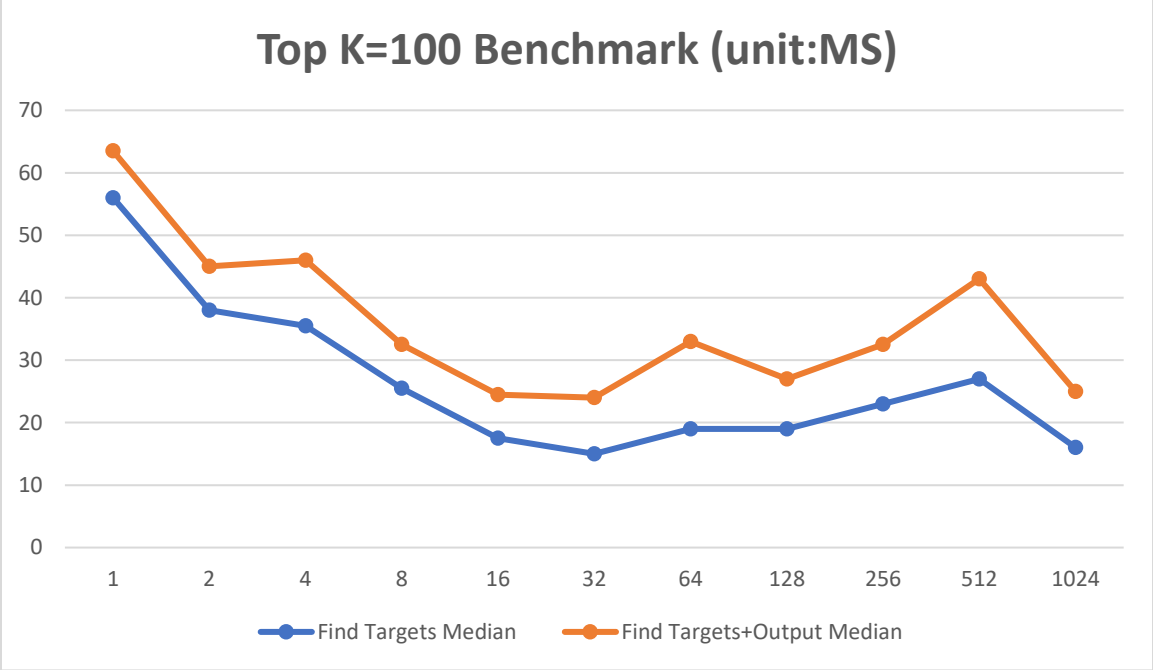
	1	2	4	8	16	32	64	128	256	512	1024
Find MIN	39	21	21	18	17	9	8	14	20	16	14
Total MIN	41	23	23	19	19	11	9	16	22	19	17
Find Median	49.5	29.5	37	27	19	11	10.5	22	32	22.5	17.5
Total Median	52.5	32.5	41.5	28.5	25.5	14.5	15.5	30	38	26	23.5
Find MAX	140	83	123	145	78	140	1854	74	96	62	744
Total MAX	185	108	127	155	84	152	1862	80	663	70	1650

Find-Find Targets

Total-Find Targets + Output

Top K=100 Benchmark (unit:MS)

Case: 65 million records find 100 records based on the top K condition



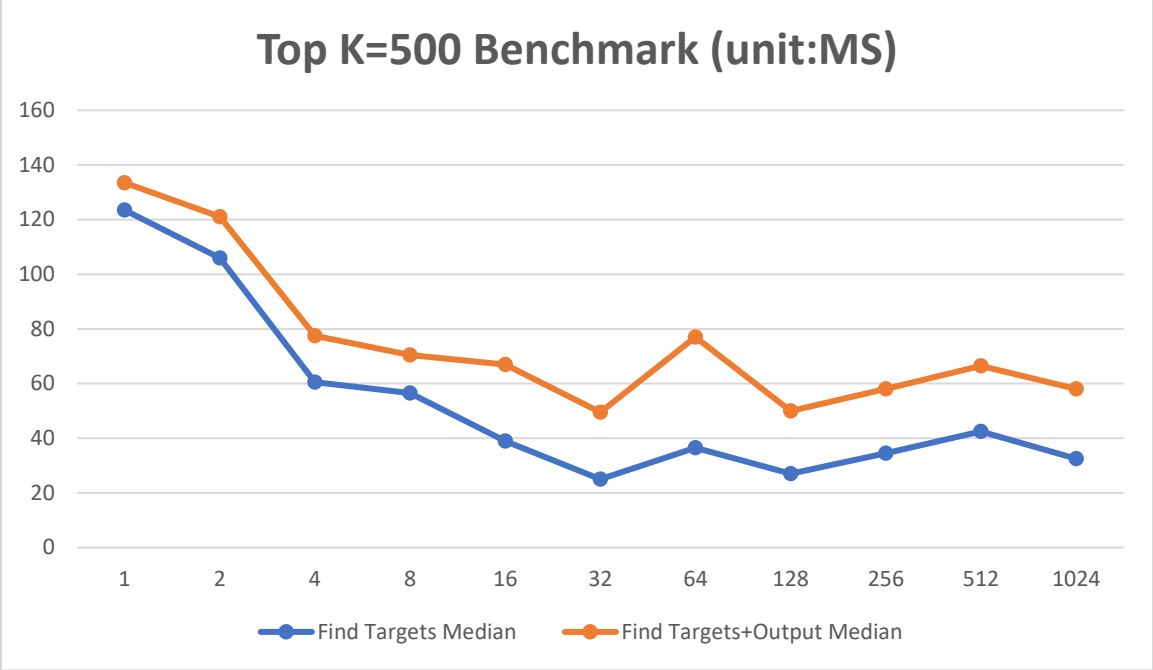
	1	2	4	8	16	32	64	128	256	512	1024
Find MIN	45	27	22	21	15	10	13	12	20	22	14
Total MIN	47	30	30	24	20	13	18	16	26	27	19
Find Median	56	38	35.5	25.5	17.5	15	19	19	23	27	16
Total Median	63.5	45	46	32.5	24.5	24	33	27	32.5	43	25
Find MAX	128	83	109	617	161	3068	98	135	90	1357	72
Total MAX	190	123	162	662	203	3168	150	184	128	1853	115

Find-Find Targets

Total-Find Targets + Output

Top K=500 Benchmark (unit:MS)

Case: 65 million records find 500 records based on the top K condition



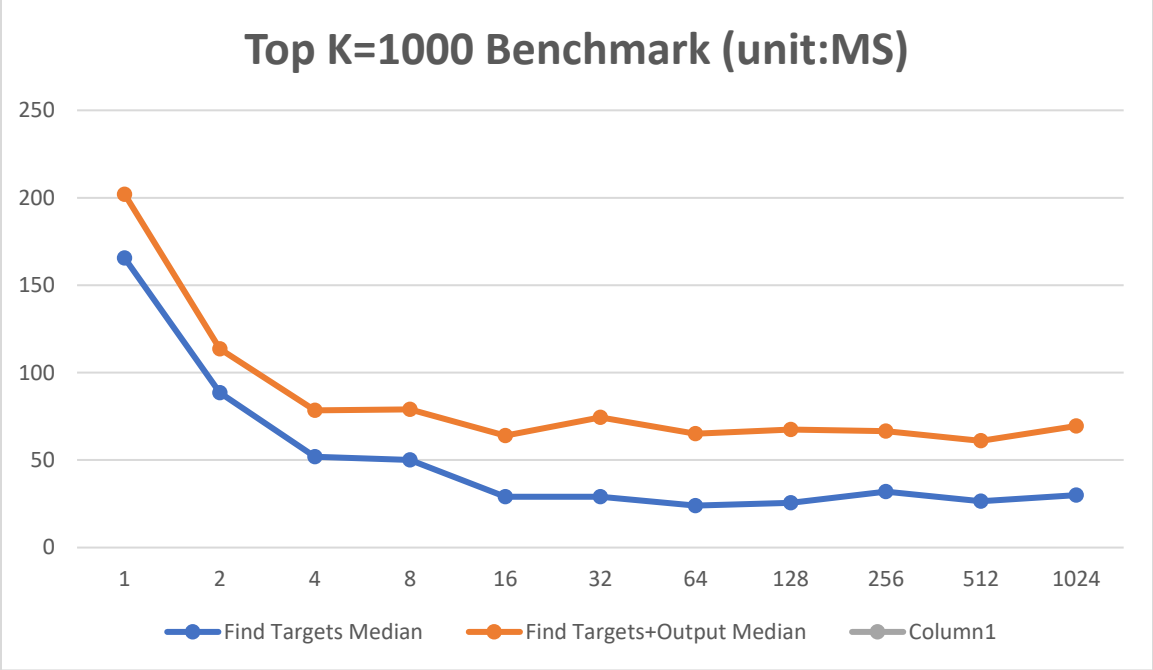
	1	2	4	8	16	32	64	128	256	512	1024
Find MIN	115	89	56	43	32	22	20	22	30	31	25
Total MIN	128	107	67	56	53	43	35	42	49	55	45
Find Median	123.5	106	60.5	56.5	39	25	36.5	27	34.5	42.5	32.5
Total Median	133.5	121	77.5	70.5	67	49.5	77	50	58	66.5	58
Find MAX	241	293	153	155	280	398	408	1276	795	198	1047
Total MAX	951	443	301	269	1102	532	1391	2009	927	755	1164

Find-Find Targets

Total-Find Targets + Output

Top K=1000 Benchmark (unit:MS)

Case: 65 million records find 1000 records based on the top K condition



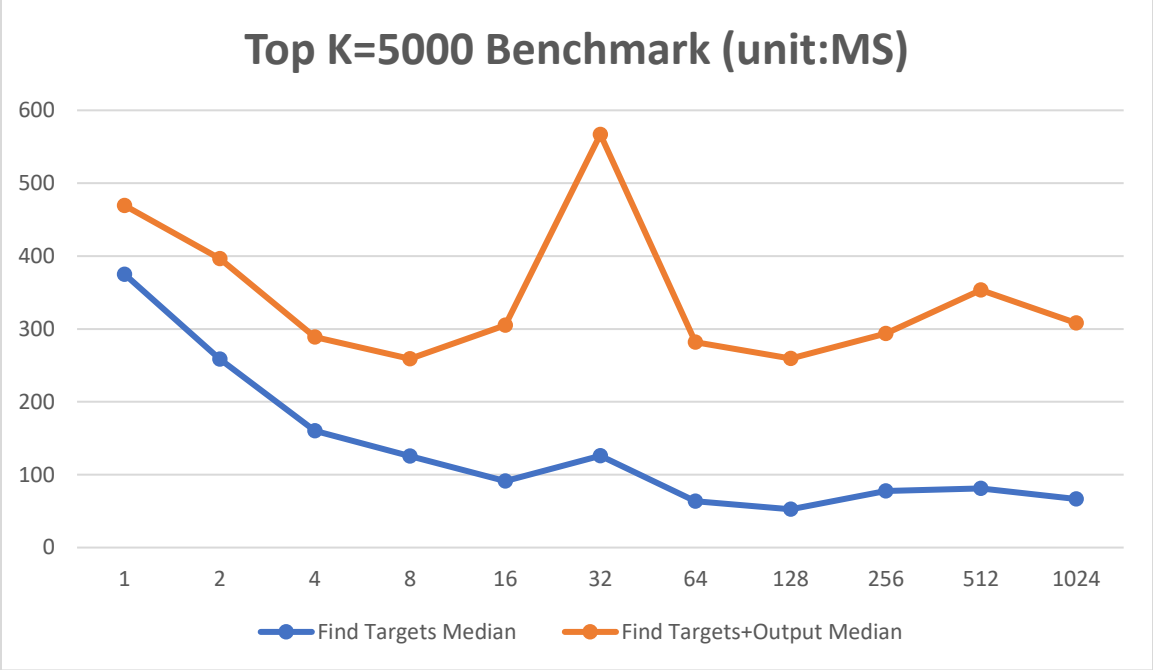
	1	2	4	8	16	32	64	128	256	512	1024
Find MIN	128	80	50	44	27	21	20	21	27	24	27
Total MIN	154	107	74	69	54	61	56	51	56	54	61
Find Median	165.5	88.5	52	50	29	29	24	25.5	32	26.5	30
Total Median	202	113.5	78.5	79	64	74.5	65	67.5	66.5	61	69.5
Find MAX	1459	282	511	2812	1802	583	166	135	1119	181	2802
Total MAX	1700	1024	1442	8169	2577	825	1021	343	2102	1350	7342

Find-Find Targets

Total-Find Targets + Output

Top K=5000 Benchmark (unit:MS)

Case: 65 million records find 5000 records based on the top K condition



	1	2	4	8	16	32	64	128	256	512	1024
Find MIN	341	223	147	105	69	73	54	37	54	62	56
Total MIN	442	359	262	222	227	345	233	234	211	265	243
Find Median	375	258.5	160	125.5	91	126	63.5	52.5	77.5	81	66.5
Total Median	469.5	396.5	289	259	305	567	282	259.5	293.5	353.5	308
Find MAX	843	646	5007	1330	1473	741	4211	1496	2076	550	726
Total MAX	15603	1984	10655	3603	3356	2949	6366	6485	4826	2659	2366

Find-Find Targets

Total-Find Targets + Output

Big Data Benchmark in Linux System

Benchmark Experiment Description

Hardware:

Memory: 64 GiB

Processor: Intel® Xeon(R) CPU E5-2630 v4 @ 2.20GHz × 40

OS Type: 64-bit

System: Linux (Fedora)

Disk: HDD

Performance Test Dataset:

PUDG Game Data: 65 Million Records, 9.00 GB

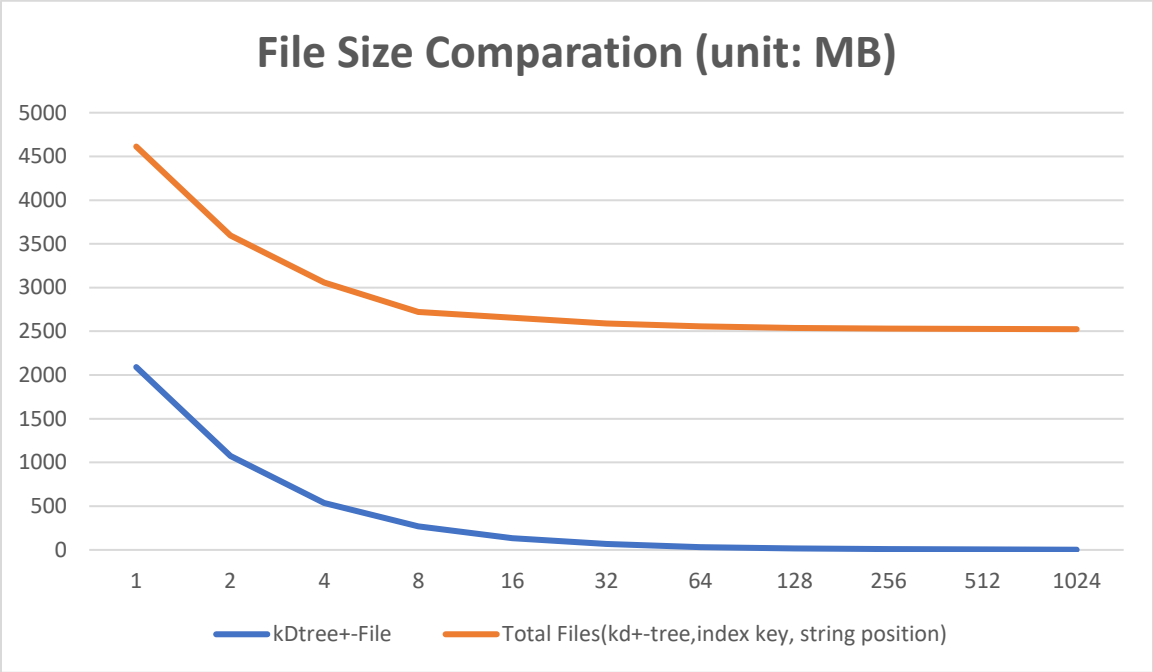
Performance Program:

Java version KD+-Tree

Performance Comparation Tasks:

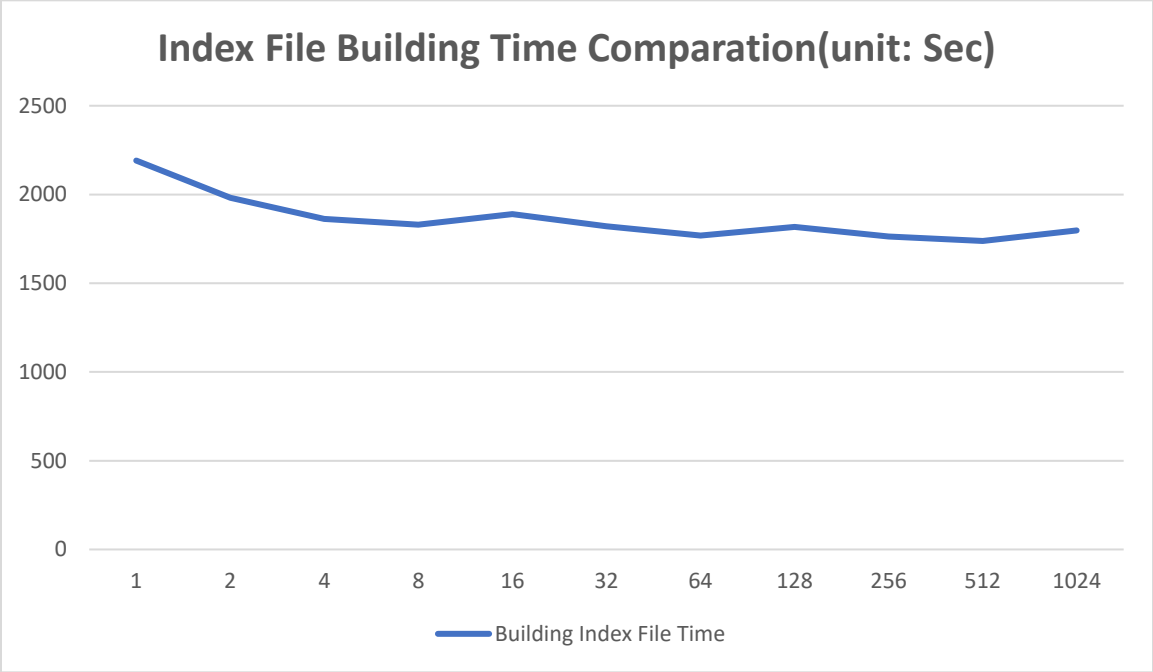
- Index files size benchmark as the knode size is different
- Building Index files time benchmark as the knode size is different
- PUDG data range search cost time
- PUDG data topK nearest POI search cost time as K = 1,10,100,500,1000,5000

File Size Comparation (unit: MB)



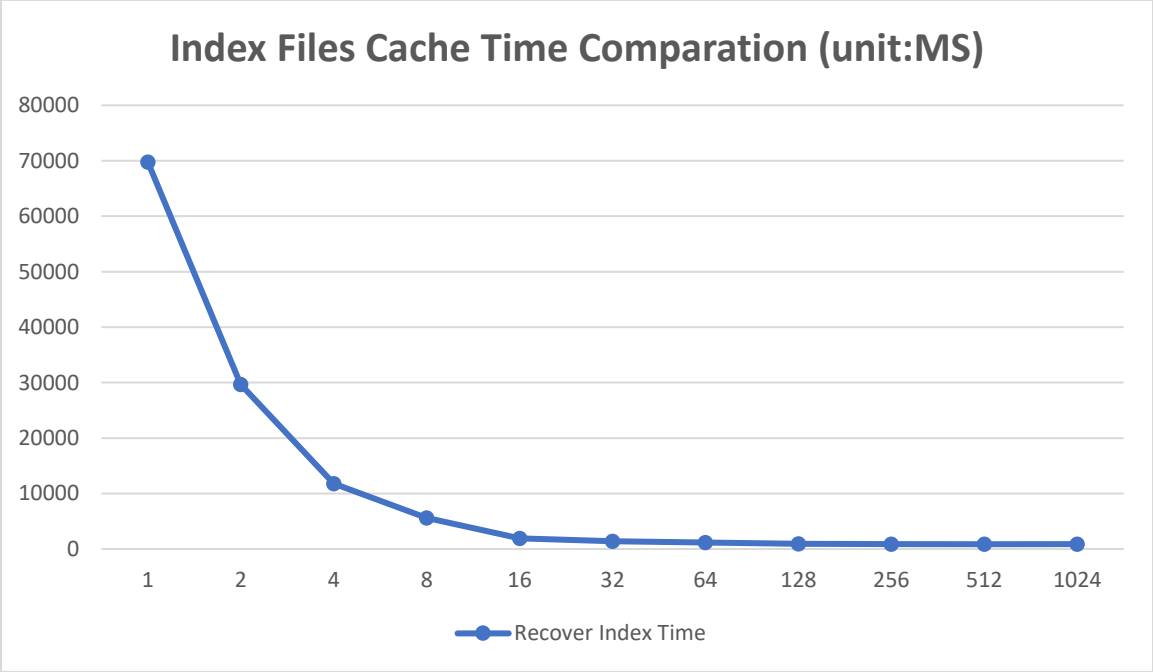
	kDtree+-File	Total Files(kD+-tree,index key, string position)
1	2091	4613
2	1073	3595
4	537	3059
8	268	2720
16	134	2656
32	67	2589
64	33	2555
128	16	2538
256	8.4	2530.4
512	4.2	2526.2
1024	2.1	2524.1

Index File Building Time Comparation (unit: SEC)



	Building Index File Time
1	2191
2	1981
4	1862
8	1829
16	1890
32	1820
64	1768
128	1817
256	1764
512	1738
1024	1798

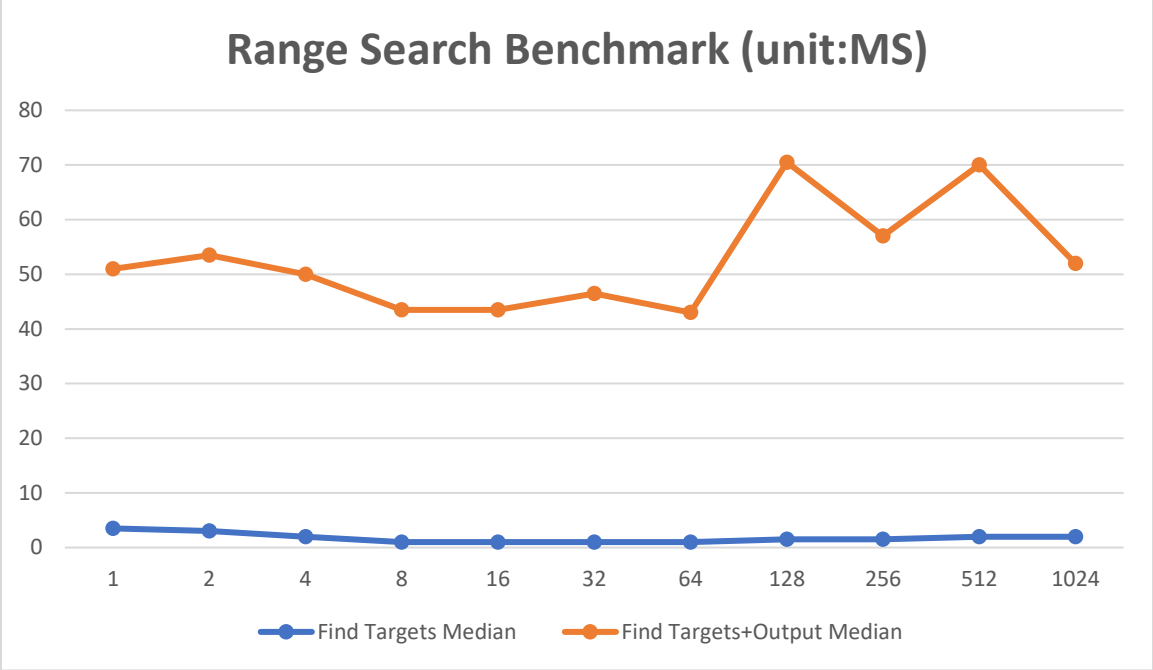
Index Files Cache Time Comparation (unit:MS)



	Cache Index Time
1	69774
2	29696
4	11792
8	5616
16	1938
32	1402
64	1156
128	920
256	914
512	867
1024	910

Range Search Benchmark (unit:MS)

Case: 65 million records find 1792 records based on the range condition



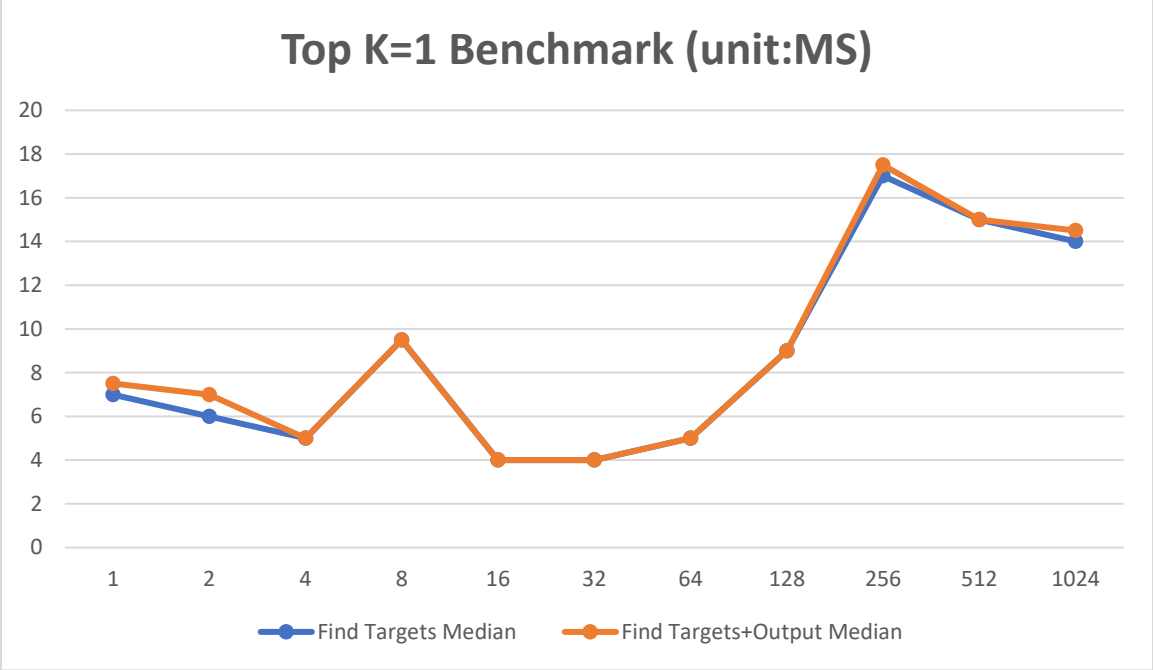
	1	2	4	8	16	32	64	128	256	512	1024
Find MIN	3	2	1	1	0	1	0	1	0	1	1
Total MIN	5	44	43	31	35	32	40	42	41	44	36
Find Median	3.5	3	2	1	1	1	1	1.5	1.5	2	2
Total Median	51	53.5	50	43.5	43.5	46.5	43	70.5	57	70	52
Find MAX	6	6	6	5	5	5	22	28	40	37	32
Total MAX	90	99	5874	3061	186	9716	12352	12281	11868	12208	12108

Find-Find Targets

Total-Find Targets + Output

Top K=1 Benchmark (unit:MS)

Case: 65 million records find 1 record based on the top K condition



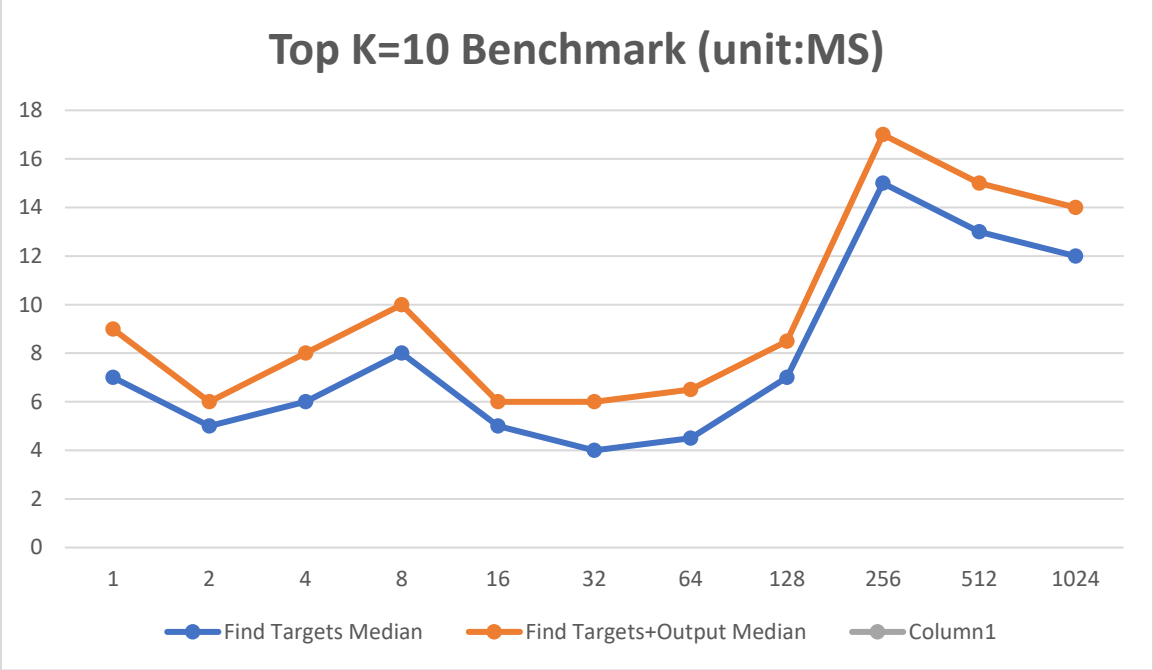
	1	2	4	8	16	32	64	128	256	512	1024
Find MIN	7	5	4	6	3	3	4	6	12	14	13
Total MIN	7	5	5	6	4	3	4	6	12	14	14
Find Median	7	6	5	9.5	4	4	5	9	17	15	14
Total Median	7.5	7	5	9.5	4	4	5	9	17.5	15	14.5
Find MAX	19	15	19	28	163	150	143	157	166	160	168
Total MAX	21	17	21	30	180	156	152	169	175	166	179

Find-Find Targets

Total-Find Targets + Output

Top K=10 Benchmark (unit:MS)

Case: 65 million records find 10 records based on the top K condition



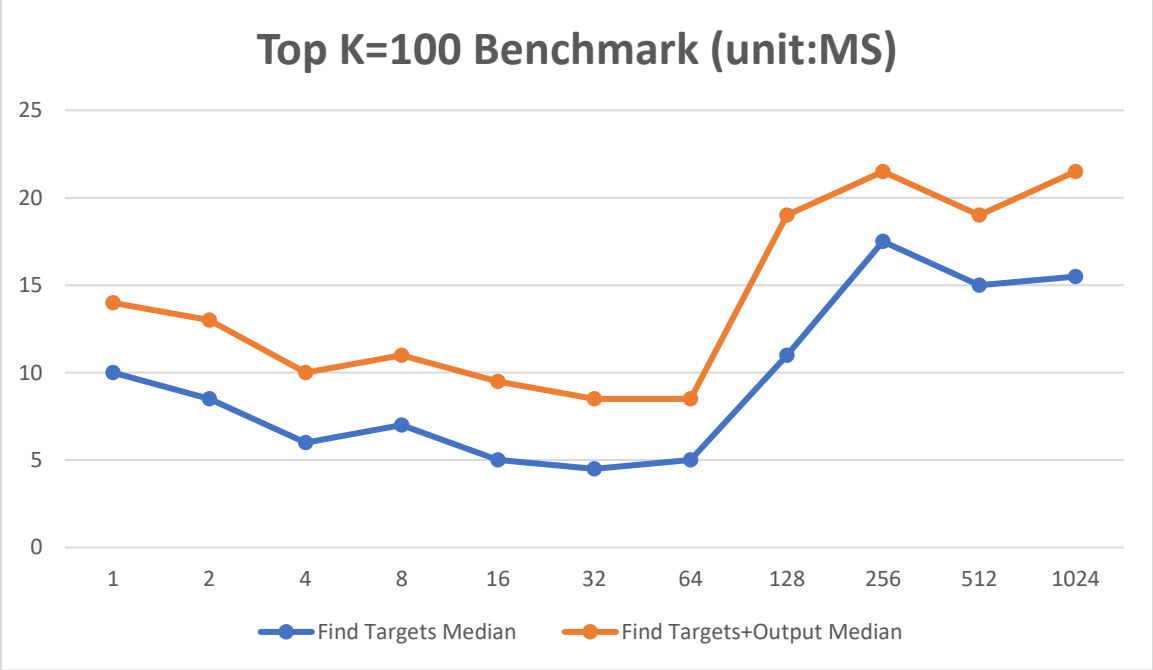
	1	2	4	8	16	32	64	128	256	512	1024
Find MIN	6	4	4	7	4	3	3	6	14	12	11
Total MIN	7	5	5	8	5	4	4	7	15	13	11
Find Median	7	5	6	8	5	4	4.5	7	15	13	12
Total Median	9	6	8	10	6	6	6.5	8.5	17	15	14
Find MAX	19	19	20	24	29	20	25	26	35	33	33
Total MAX	300	137	105	120	121	121	97	117	116	247	102

Find-Find Targets

Total-Find Targets + Output

Top K=100 Benchmark (unit:MS)

Case: 65 million records find 100 records based on the top K condition



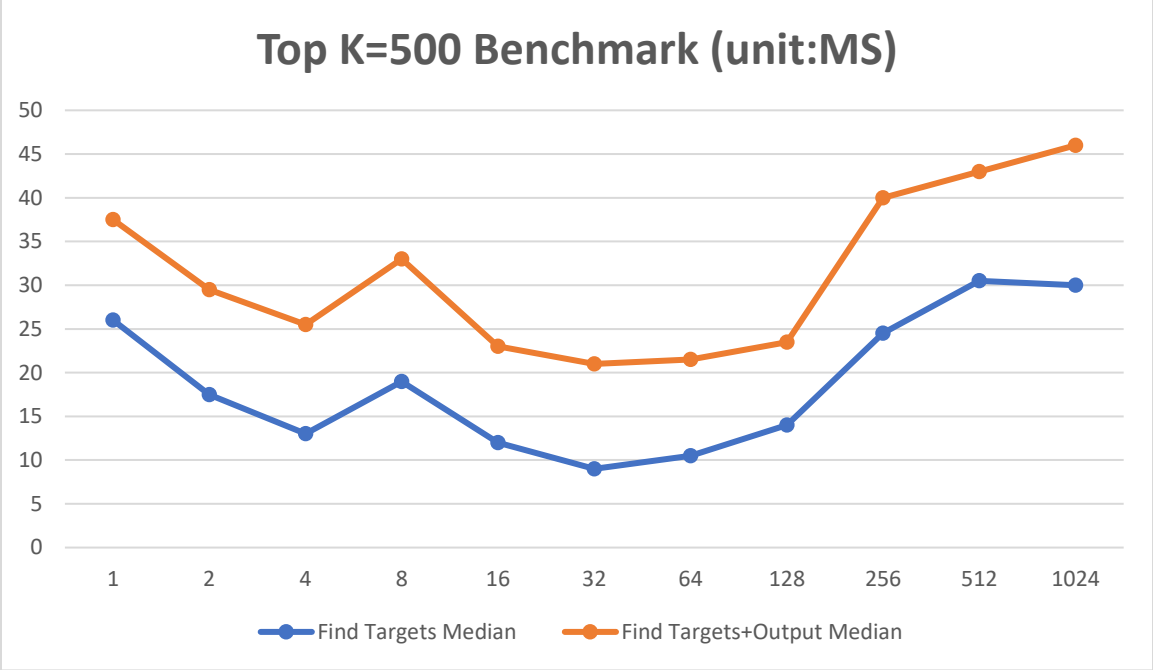
	1	2	4	8	16	32	64	128	256	512	1024
Find MIN	8	6	6	7	4	3	4	9	12	12	15
Total MIN	10	9	8	10	6	5	6	12	15	14	18
Find Median	10	8.5	6	7	5	4.5	5	11	17.5	15	15.5
Total Median	14	13	10	11	9.5	8.5	8.5	19	21.5	19	21.5
Find MAX	319	96	71	63	69	104	78	86	88	122	98
Total MAX	948	744	6345	714	710	676	728	704	693	801	769

Find-Find Targets

Total-Find Targets + Output

Top K=500 Benchmark (unit:MS)

Case: 65 million records find 500 records based on the top K condition



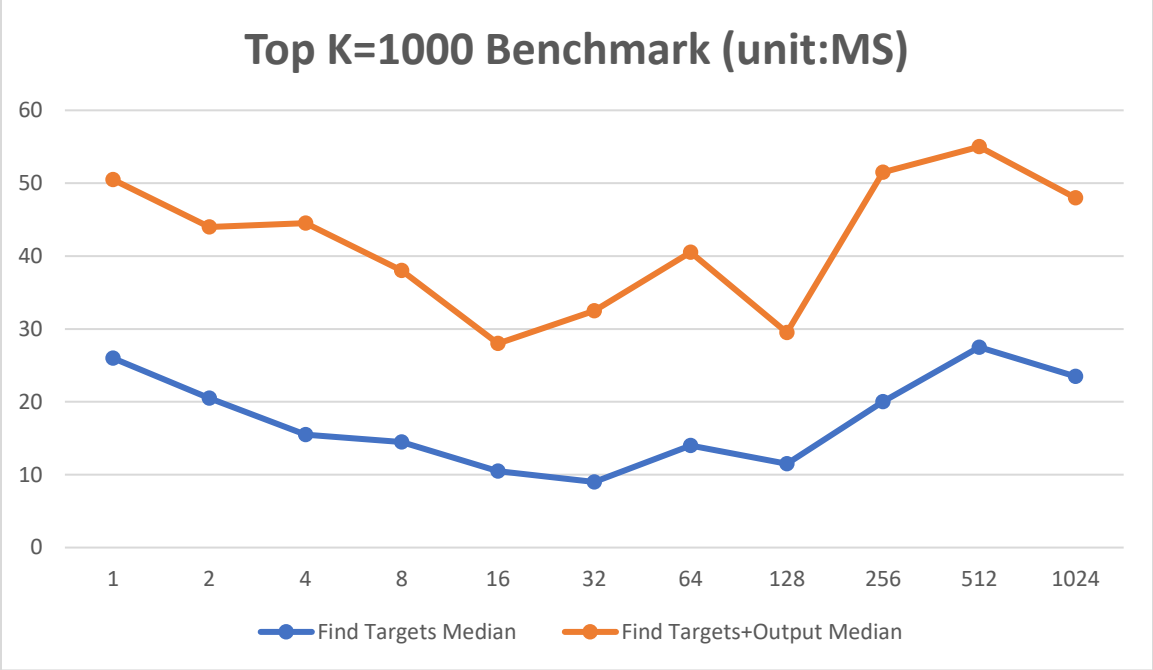
	1	2	4	8	16	32	64	128	256	512	1024
Find MIN	22	15	11	14	10	8	8	9	15	15	19
Total MIN	31	25	22	26	20	1	17	18	24	23	31
Find Median	26	17.5	13	19	12	9	10.5	14	24.5	30.5	30
Total Median	37.5	29.5	25.5	33	23	21	21.5	23.5	40	43	46
Find MAX	377	217	173	2978	195	205	180	158	189	187	200
Total MAX	3263	2923	8881	2990	3087	2859	2947	2895	2948	3168	3180

Find-Find Targets

Total-Find Targets + Output

Top K=1000 Benchmark (unit:MS)

Case: 65 million records find 1000 records based on the top K condition



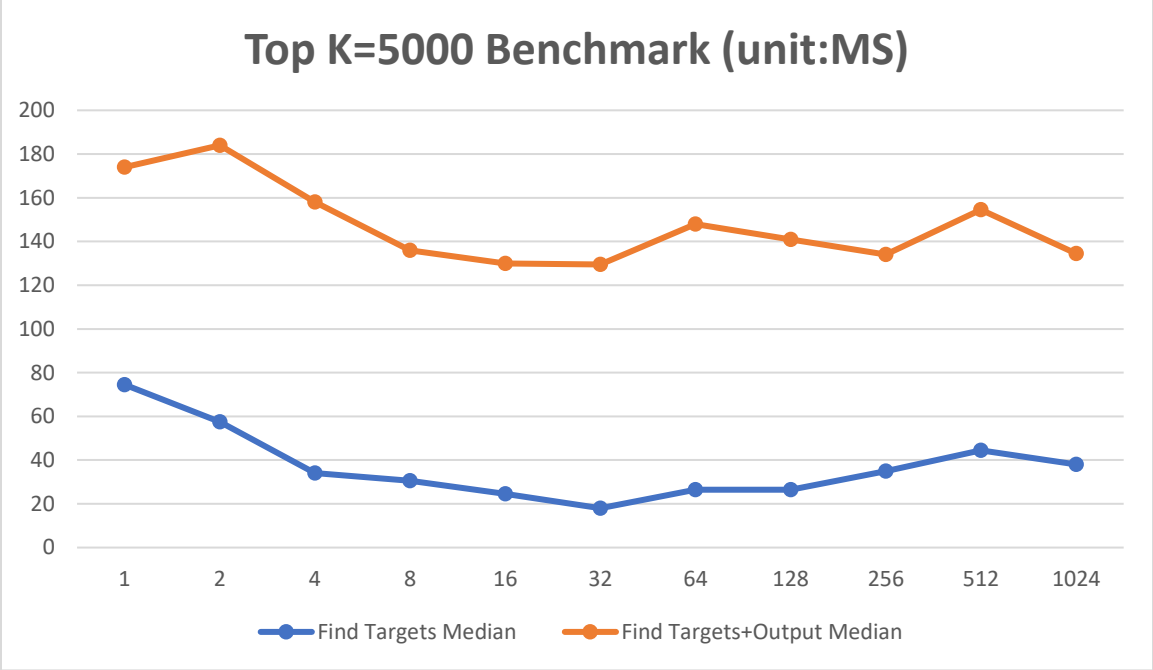
	1	2	4	8	16	32	64	128	256	512	1024
Find MIN	23	18	13	12	8	8	8	9	15	20	18
Total MIN	44	41	35	31	23	28	27	22	34	39	36
Find Median	26	20.5	15.5	14.5	10.5	9	14	11.5	20	27.5	23.5
Total Median	50.5	44	44.5	38	28	32.5	40.5	29.5	51.5	55	48
Find MAX	63	41	45	57	40	46	51	62	71	75	68
Total MAX	3791	3474	3448	3373	3506	3516	3583	3467	3671	3543	3570

Find-Find Targets

Total-Find Targets + Output

Top K=5000 Benchmark (unit:MS)

Case: 65 million records find 5000 records based on the top K condition



	1	2	4	8	16	32	64	128	256	512	1024
Find MIN	67	45	32	27	20	15	19	20	28	31	28
Total MIN	166	151	135	110	99	99	92	98	105	107	109
Find Median	74.5	57.5	34	30.5	24.5	18	26.5	26.5	35	44.5	38
Total Median	174	184	158	136	130	129.5	148	141	134	154.5	134.5
Find MAX	521	332	6191	223	326	1057	215	218	248	258	258
Total MAX	28338	28653	34169	31147	27913	27828	27938	28167	27513	27911	27910

Find-Find Targets

Total-Find Targets + Output