日志分析场景公开

access_log数据集

ClickHouse表结构DDL

```
CREATE TABLE access_log_local on cluster default
(
  `sql` String,
  `schema` String,
  `type` String,
  `access_ip` String,
  `conn_id` UInt32,
  `process_id` String,
  `logic_ins_id` UInt32,
  `accept_time` UInt64,
  `_date` DateTime,
  `total_time` UInt32,
  `succeed` String,
  `inst_name` String
)
ENGINE = MergeTree()
PARTITION BY toYYYYMMDD(_date)
```

```
ORDER BY (logic_ins_id, accept_time);

CREATE TABLE access_log on cluster default as access_log_local
engine = Distributed(default, default, access_log_local, rand());
```

Elasticsearch表结构DDL

```
PUT /access_log?pretty
{
  "settings": {
    "index": {
      "number_of_shards": 8,
      "number_of_replicas": 1,
      "max_result_window": 1000000,
      "max_inner_result_window": 10000,
      "requests.cache.enable": false,
      "queries.cache.enabled": false,
      "warmer.enabled": false,
      "sort.field": [
        "logic_ins_id",
        "accept_time"
      ]
    }
  },
  "mappings": {
    "properties": {
      "_date": {
        "type": "date",
        "format": "yyyy-MM-dd HH:mm:ss||yyyy-MM-dd||epoch_millis"
      },
      "accept_time": {
        "type": "long"
      },
      "access_ip": {
        "type": "keyword"
      },
      "conn_id": {
        "type": "integer"
```

```
      },
      "inst_name": {
        "type": "keyword"
      },
      "logic_ins_id": {
        "type": "integer"
      },
      "process_id": {
        "type": "keyword"
      },
      "schema": {
        "type": "keyword"
      },
      "sql": {
        "type": "text"
      },
      "succeed": {
        "type": "keyword"
      },
      "total_time": {
        "type": "integer"
      },
      "type": {
        "type": "keyword"
      }
    }
  }
}
```

ClickHouse查询语句

```
--Q1
select _date, accept_time, access_ip, type, total_time, concat(toString(total_time),'ms') as total_time_ms, sql,schema,succeed,process_id,inst_name from access_log where _date >= '2020-12-27 00:38:31' and _date <= '2020-12-28 00:38:31' and logic_ins_id = 502680264 and accept_time <= 1609087111000 and accept_time >= 16090007
```

```
11000 and positionCaseInsensitive(sql, 'select') > 0 order by accept_time desc limit 50,50;
--Q2
select
case
when total_time <=100 then 1
when total_time > 100 and total_time <= 500 then 2
when total_time > 500 and total_time <= 1000 then 3
when total_time > 1000 and total_time <= 3000 then 4
when total_time > 3000 and total_time <= 10000 then 5
when total_time > 10000 and total_time <= 30000 then 6
else 7
end as reorder,
case
when total_time <=100 then '0~100ms'
when total_time > 100 and total_time <= 500 then '100ms~500ms'
when total_time > 500 and total_time <= 1000 then '500ms~1s'
when total_time > 1000 and total_time <= 3000 then '1s~3s'
when total_time > 3000 and total_time <= 10000 then '3s~10s'
when total_time > 10000 and total_time <= 30000 then '10s~30s'
else '30s以上'
end as label,
case
when total_time <= 100 then '0~100'
when total_time > 100 and total_time <= 500 then '100~500'
when total_time > 500 and total_time <= 1000 then '500~1000'
when total_time > 1000 and total_time <= 3000 then '1000~3000'
when total_time > 3000 and total_time <= 10000 then '3000~10000'
when total_time > 10000 and total_time <= 30000 then '10000~30000'
else '30000~10000000000'
end as vlabel,
count() as value
from access_log
where logic_ins_id = 502867976 and _date >= '2020-12-27 00:38:31' and _date <= '2020-12-28 00:38:31' and accept_time <= 1609087111000 and accept_time >= 1609000711000
group by label,vlabel,reorder
order by reorder;
--Q3
```

```
select toStartOfMinute(_date) as time, count() as value
from access_log
where logic_ins_id = 500152868 and accept_time <= 1609087111000 and accept_time >= 1609000711000
group by time
order by time;
--Q4
select count(*) as c from (
  select _date, accept_time, access_ip, type, total_time, concat(toString(total_time),'ms') as total_time_ms, sql, schema, succeed, process_id, inst_name
  from access_log
  where logic_ins_id = 501422856 and _date >= '2020-12-27 00:38:31' and _date <= '2020-12-28 00:38:31' and accept_time <= 1609087111000 and accept_time >= 1609000711000
);
```

Elasticsearch查询语句

```
--Q1
POST /_sql?format=txt&pretty
{
  "query": "select _date, accept_time, access_ip, type, total_time, concat(cast(total_time as string),'ms') as total_time_ms, sql, schema,succeed,process_id,inst_name from access_log where _date >= '2020-12-27 00:38:31' and _date <= '2020-12-28 00:38:31' and logic_ins_id = 502680264 and accept_time <= 1609087111000 and accept_time >= 1609000711000 and match(sql, 'select') order by accept_time desc limit 100"
}

--Q2
POST /_sql?format=txt&pretty
{
  "query": "select case when total_time <=100 then 1 when total_time > 100 and total_time <= 500 then 2 when total_time > 500 and total_time <= 1000 then 3 when total_time > 1000 and total_time <= 3000 then 4 when total_time > 3000 and total_time <= 10000 th
```

```
en 5 when total_time > 10000 and total_time <= 30000 then 6 else
 7 end as reorder, case when total_time <=100 then '0~100ms' when
total_time > 100 and total_time <= 500 then '100ms~500ms' when to
tal_time > 500 and total_time <= 1000 then '500ms~1s' when total_
time > 1000 and total_time <= 3000 then '1s~3s' when total_time >
3000 and total_time <= 10000 then '3s~10s' when total_time > 1000
0 and total_time <= 30000 then '10s~30s' else '30s以上' end as lab
el, case when total_time <= 100 then '0~100' when total_time > 10
0 and total_time <= 500 then '100~500' when total_time > 500 and
 total_time <= 1000 then '500~1000' when total_time > 1000 and to
tal_time <= 3000 then '1000~3000' when total_time > 3000 and tota
l_time <= 10000 then '3000~10000' when total_time > 10000 and tot
al_time <= 30000 then '10000~30000' else '30000~10000000000' end
 as vlabel, count(*) as value from access_log where logic_ins_id
 = 502867976 and _date >= '2020-12-27 00:38:31' and _date <= '202
0-12-28 00:38:31' and accept_time <= 1609087111000 and accept_tim
e >= 1609000711000 group by label,vlabel,reorder order by reorde
r"
}

--Q3
POST /_sql?format=txt&pretty
{
  "query": "select date_trunc('mi', _date) as time, count(*) as v
alue from access_log where logic_ins_id = 500152868 and accept_ti
me <= 1609087111000 and accept_time >= 1609000711000 group by 1 o
rder by 1"
}

--Q4
POST /_sql?format=txt&pretty
{
  "query": "select count(*) as c from (select _date, accept_time,
access_ip, type, total_time, concat(cast(total_time as string),'m
s') as total_time_ms, sql, schema, succeed, process_id, inst_name
from access_log where logic_ins_id = 501422856 and _date >= '2020
-12-27 00:38:31' and _date <= '2020-12-28 00:38:31' and accept_ti
me <= 1609087111000 and accept_time >= 1609000711000)"
}
```

trace_log数据集

ClickHouse表结构DDL

```
CREATE TABLE trace_local on cluster default
(
  `serviceName` LowCardinality(String),
  `host` LowCardinality(String),
  `ip` String,
  `spanName` String,
  `spanId` String,
  `pid` LowCardinality(String),
  `parentSpanId` String,
  `ppid` String,
  `duration` Int64,
  `rpcType` Int32,
  `startTime` Int64,
  `traceId` String,
  `tags.k` Array(String),
  `tags.v` Array(String),
  `events` String,
  KEY trace_idx traceId TYPE range
) ENGINE = MergeTree()
PARTITION BY intDiv(startTime, toInt64(7200000000))
PRIMARY KEY (serviceName, host, ip, pid, spanName)
ORDER BY (serviceName, host, ip, pid, spanName, tags.k);

CREATE TABLE trace on cluster default as trace_local
engine = Distributed(default, default, trace_local, rand());
```

Elasticsearch表结构DDL

```
PUT /trace?pretty
{
  "settings" : {
    "index" : {
      "number_of_shards" : 32,
```

```
      "number_of_replicas" : 1,
      "max_result_window" : 1000000,
      "max_inner_result_window" : 10000,
      "requests.cache.enable": false,
      "queries.cache.enabled": false,
      "warmer.enabled": false,
      "sort.field": [ "serviceName", "host", "ip", "pid", "spanName", "tags.k" ]
    }
  },
  "mappings" : {
    "properties" : {
      "duration" : {
        "type" : "long"
      },
      "events" : {
        "type" : "keyword"
      },
      "host" : {
        "type" : "keyword"
      },
      "ip" : {
        "type" : "keyword"
      },
      "parentSpanId" : {
        "type" : "keyword"
      },
      "pid" : {
        "type" : "keyword"
      },
      "ppid" : {
        "type" : "keyword"
      },
      "rpcType" : {
        "type" : "integer"
      },
      "serviceName" : {
        "type" : "keyword"
      },
      "spanId" : {
```

```
        "type" : "keyword"
      },
      "spanName" : {
        "type" : "keyword"
      },
      "startTime" : {
        "type" : "long"
      },
      "tags" : {
        "properties" : {
          "k" : {
            "type" : "keyword"
          },
          "v" : {
            "type" : "keyword"
          }
        }
      },
      "traceId" : {
        "type" : "keyword"
      }
    }
  }
}
```

ClickHouse查询语句

```
--Q1
select *
from trace
prewhere
traceId ='ccc6084420b76183'
where startTime > 1597968000300000  and startTime <  1598054399099000 settings max_threads = 1;
--Q2
select count(*) count, spanName as name from trace
where serviceName ='conan-dean-user-period'
and startTime > 1597968000300000  and startTime <  15980543990990
```

```
00
group by spanName
order by count desc limit 1000;
--Q3
select host as name, count(*) count
from trace
where serviceName ='conan-dean-user-period'
and startTime > 1597968000300000  and startTime <  1598054399099000
group by host;
--Q4
select count(*) count, tags.k as name  from trace
array join tags.k
where serviceName ='conan-dean-user-period'
and startTime > 1597968000300000  and startTime <  1598054399099000
group by tags.k;
--Q5
select count(*) spancount,
sum(duration) as sumDuration, intDiv(startTime, 1440000000) as timeSel
from trace
where serviceName ='conan-dean-user-period'
and startTime > 1597968000300000  and startTime <  1598054399099000
group by timeSel;
--Q6
select count(*) spanCount,
countIf(duration  <=1000000), countIf(duration > 1000000),  countIf(duration > 3000000)
from trace
where serviceName ='conan-dean-user-period'
and startTime > 1597968000300000  and startTime <  1598054399099000;
--Q7
select  host, startTime,traceId,spanName,duration,tags.k,tags.v
from trace
where serviceName ='conan-dean-user-period'
and startTime > 1597968000300000  and startTime <  1598054399099000 limit 1000000;
```

Elasticsearch查询语句

```
--Q1
GET /trace/_search?pretty
{
  "size" : 1000,
  "query" : {
    "bool" : {
      "must" : [
        {
          "term" : {
            "traceId" : {
              "value" : "ccc6084420b76183",
              "boost" : 1.0
            }
          }
        },
        {
          "range" : {
            "startTime" : {
              "from" : 1597968000300000,
              "to" : 1598054399099000,
              "include_lower" : false,
              "include_upper" : false,
              "time_zone" : "Z",
              "boost" : 1.0
            }
          }
        }
      ],
      "adjust_pure_negative" : true,
      "boost" : 1.0
    }
  },
  "sort" : [
    {
      "_doc" : {
```

```
        "order" : "asc"
      }
    }
  ]
}

--Q2
POST /_sql?format=txt&pretty
{
  "query": "select count(*) count, spanName as name from trace where serviceName ='conan-dean-user-period' and startTime > 1597968000300000  and startTime <  1598054399099000 group by spanName order by count desc limit 1000"
}

--Q3
POST /_sql?format=txt&pretty
{
  "query": "select host as name, count(*) count from trace where serviceName ='conan-dean-user-period' and startTime > 1597968000300000  and startTime <  1598054399099000 group by host"
}

--Q4
POST /_sql?format=txt&pretty
{
  "query": "select count(*) count, tags.k as name  from trace where serviceName ='conan-dean-user-period' and startTime > 1597968000300000  and startTime <  1598054399099000 group by tags.k"
}

--Q5
POST /_sql?format=txt&pretty
{
  "query": "select count(*) spancount, sum(duration) as sumDuration, (startTime / 1440000000) as timeSel from trace where serviceName ='conan-dean-user-period' and startTime > 1597968000300000 and startTime <  1598054399099000 group by timeSel"
}

```

```
--Q6
GET /trace/_search
{
  "size" : 0,
  "query" : {
    "bool" : {
      "must" : [
        {
          "term" : {
            "serviceName" : {
              "value" : "conan-dean-user-period",
              "boost" : 1.0
            }
          }
        },
        {
          "range" : {
            "startTime" : {
              "from" : 1597968000300000,
              "to" : 1598054399099000,
              "include_lower" : false,
              "include_upper" : false,
              "time_zone" : "Z",
              "boost" : 1.0
            }
          }
        }
      ],
      "adjust_pure_negative" : true,
      "boost" : 1.0
    }
  },
  "_source" : false,
  "stored_fields" : "_none_",
  "track_total_hits" : 2147483647,
  "aggregations" : {
    "groupby" : {
      "filters" : {
        "filters" : [
          {
```

```
106             "match_all" : {
107               "boost" : 1.0
108             }
109           }
110         ],
111         "other_bucket" : false,
112         "other_bucket_key" : "_other_"
113       },
114       "aggregations" : {
115         "2227391e" : {
116           "sum" : {
117             "script" : {
118               "source" : "InternalSqlScriptUtils.cast(InternalQl
    ScriptUtils.lte(InternalQlScriptUtils.docValue(doc,params.v0),pa
    rams.v1),params.v2)",
119               "lang" : "painless",
120               "params" : {
121                 "v0" : "duration",
122                 "v1" : 1000000,
123                 "v2" : "INTEGER"
124               }
125             }
126           }
127         },
128         "22265801" : {
129           "sum" : {
130             "script" : {
131               "source" : "InternalSqlScriptUtils.cast(InternalQl
    ScriptUtils.gt(InternalQlScriptUtils.docValue(doc,params.v0),par
    ams.v1),params.v2)",
132               "lang" : "painless",
133               "params" : {
134                 "v0" : "duration",
135                 "v1" : 1000000,
136                 "v2" : "INTEGER"
137               }
138             }
139           }
140         },
141         "1838381" : {
```

```
          "sum" : {
            "script" : {
              "source" : "InternalSqlScriptUtils.cast(InternalQlScriptUtils.gt(InternalQlScriptUtils.docValue(doc,params.v0),params.v1),params.v2)",
              "lang" : "painless",
              "params" : {
                "v0" : "duration",
                "v1" : 3000000,
                "v2" : "INTEGER"
              }
            }
          }
        }
      }
    }
  }
}

--Q7
GET /trace/_search
{
  "size": 1000000,
  "query": {
    "bool": {
      "must": [
        {
          "term": {
            "serviceName": {
              "value": "conan-dean-user-period",
              "boost": 1
            }
          }
        },
        {
          "range": {
            "startTime": {
              "from": 1597968000300000,
              "to": 1598054399099000,
              "include_lower": false,
```

```
              "include_upper": false,
              "time_zone": "Z",
              "boost": 1
            }
          }
        }
      ],
      "adjust_pure_negative": true,
      "boost": 1
    }
  },
  "_source": {
    "includes": [
      "host",
      "startTime",
      "traceId",
      "spanName",
      "duration",
      "tags.k",
      "tags.v"
    ],
    "excludes": []
  },
  "sort": [
    {
      "_doc": {
        "order": "asc"
      }
    }
  ]
}
```