An Elasticsearch use case in eBay's Job performance monitor system

About me

- Software engineer at eBay's Infrastructure Data Engineering
- Focus on Spark optimization and platform building
- Contributor of Spark/Hadoop
- Worked for Meituan-Dianping and Alibaba

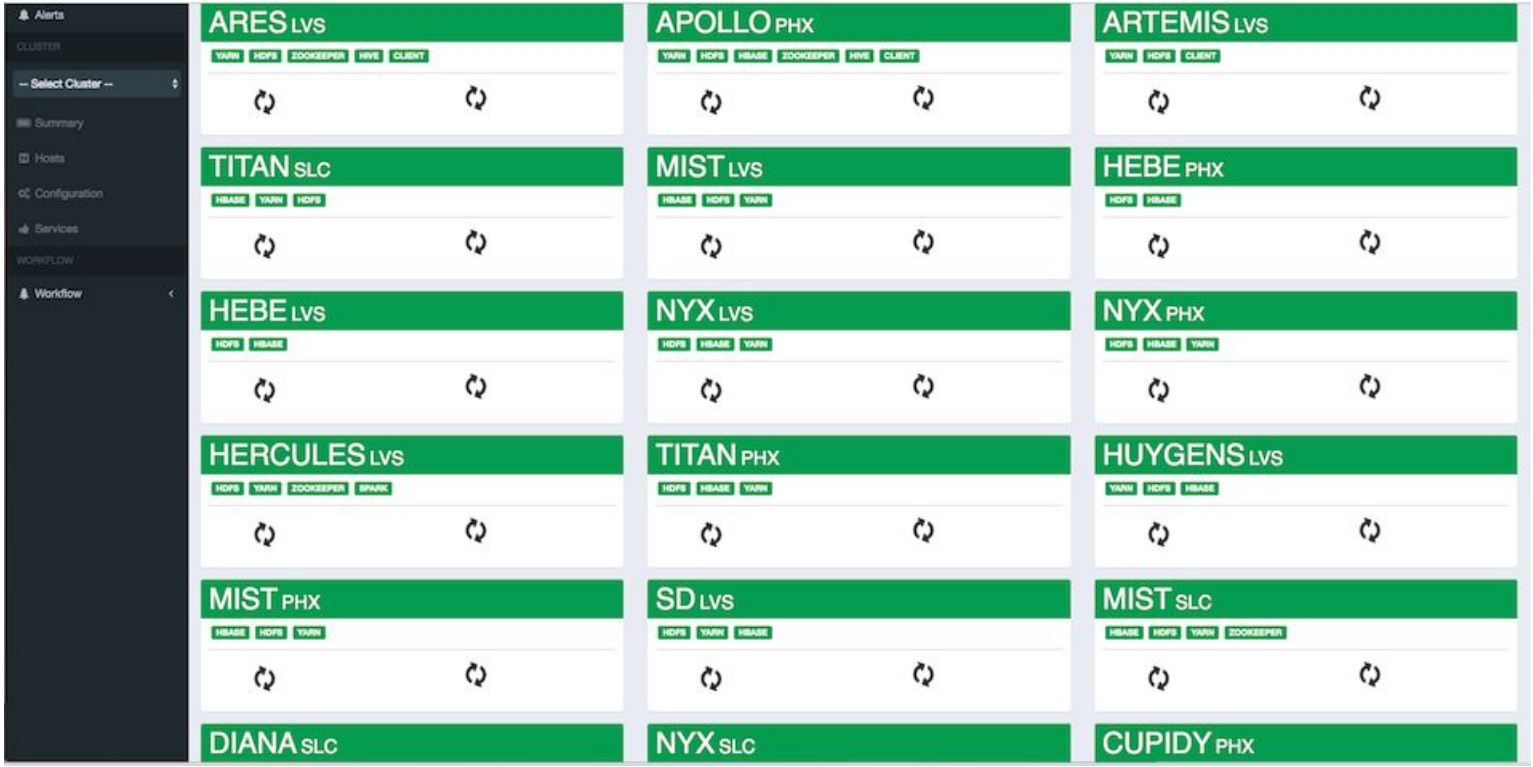
Agenda

- Introduce eBay's job performance monitor (JPM)
  - Architecture & Design
  - Success Stories
- Usage of underlying Elasticsearch
  - Object Relational Mapping
  - Rest API Extension

Introduce eBay's JPM

- Mission & Gap
- Architecture & Design
- Success Stories
- Product Demo

Challenges


Alerts
CLUSTER
-- Select Cluster --
Summary
Hosts
Configuration
Services
WORKFLOW
Workflow
ARES LVS
YARN HDFS ZOOKEEPER HIVE CLIENT
APOLLO PHX
YARN HDFS HBASE ZOOKEEPER HIVE CLIENT
ARTEMIS LVS
YARN HDFS CLIENT
TITAN SLC
HBASE YARN HDFS
MIST LVS
HBASE HDFS YARN
HEBE PHX
HDFS HBASE
HEBE LVS
HDFS HBASE
NYX LVS
HDFS HBASE YARN
NYX PHX
HDFS HBASE YARN
HERCULES LVS
HDFS YARN ZOOKEEPER SPARK
TITAN PHX
HDFS HBASE YARN
HUYGENS LVS
YARN HDFS HBASE
MIST PHX
HBASE HDFS YARN
SD LVS
HDFS YARN HBASE
MIST SLC
HBASE HDFS YARN ZOOKEEPER
DIANA SLC
NYX SLC
CUPIDY PHX

Mission

Improve Development Experience

Increase Resource Efficiency

Gaps

- Development Experience
  - Centralized logging service for failure diagnostics
  - Job/Task level metrics is hard for developer understanding
  - Application healthiness visibility
  - Tedious communication to problem resolution for any workload issue
- Resource Efficiency
  - Huge manual effort for analyze cluster/queue high load
  - Blind to “bad” jobs

Object

For Developers

- Application-specific diagnostics and Performance Recommendation
- Highlight applications need attention
- Identify bottlenecks and resource usage

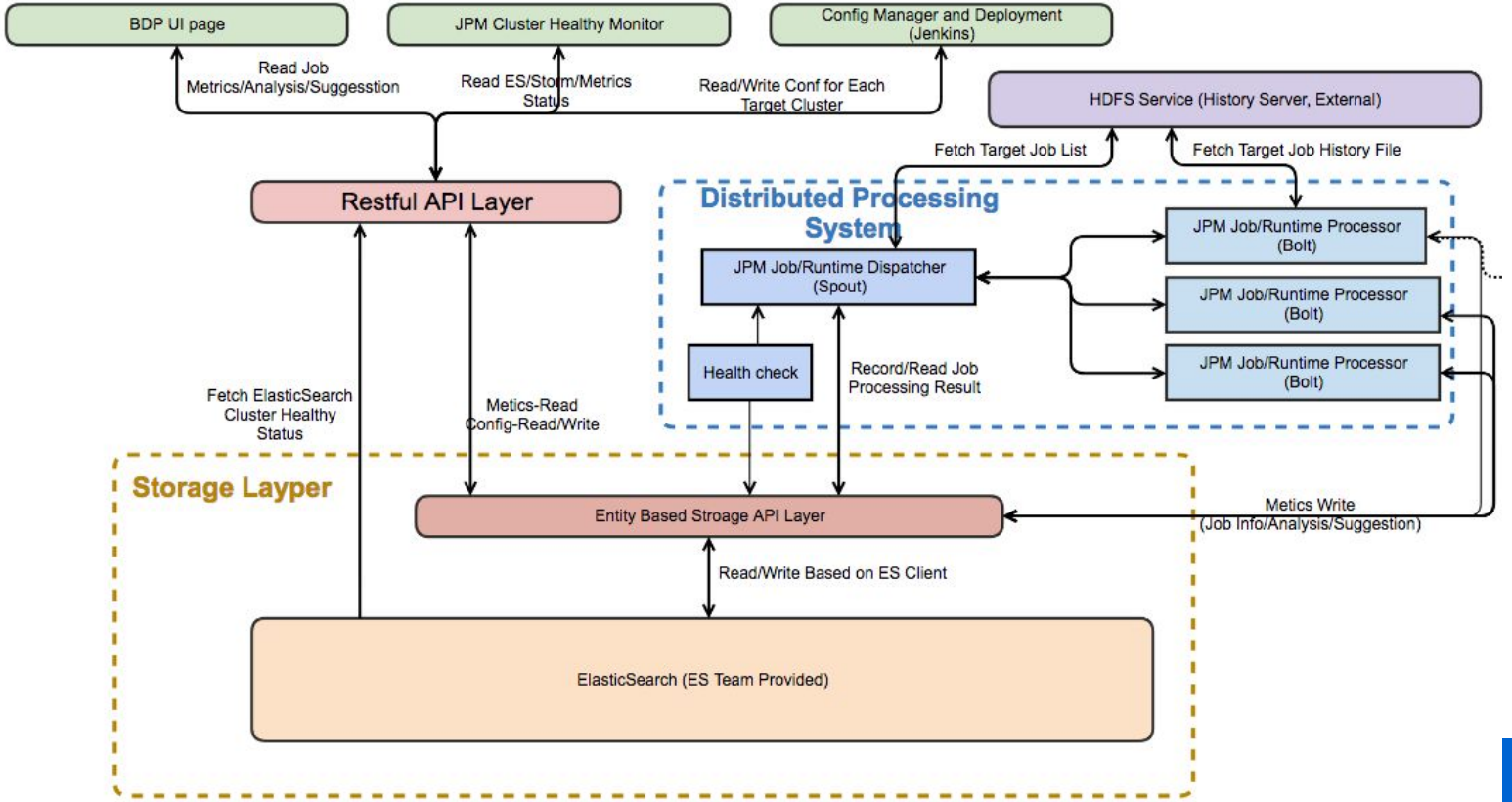
For Operators

- Reduce performance incidents in production
- Easy communication back to developer for detailed performance insights

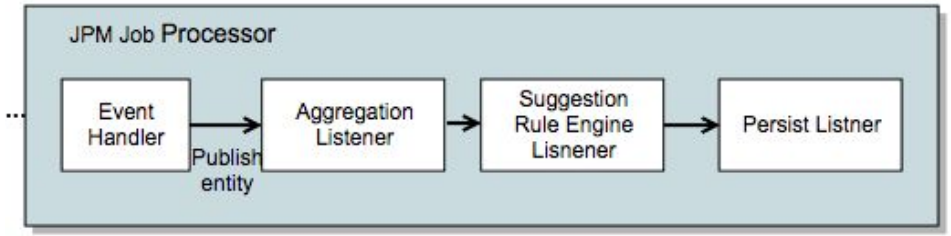
For Managers

- Shorten time to production
- Resource usage insight and guidance
- Increase cluster ROI

Architecture

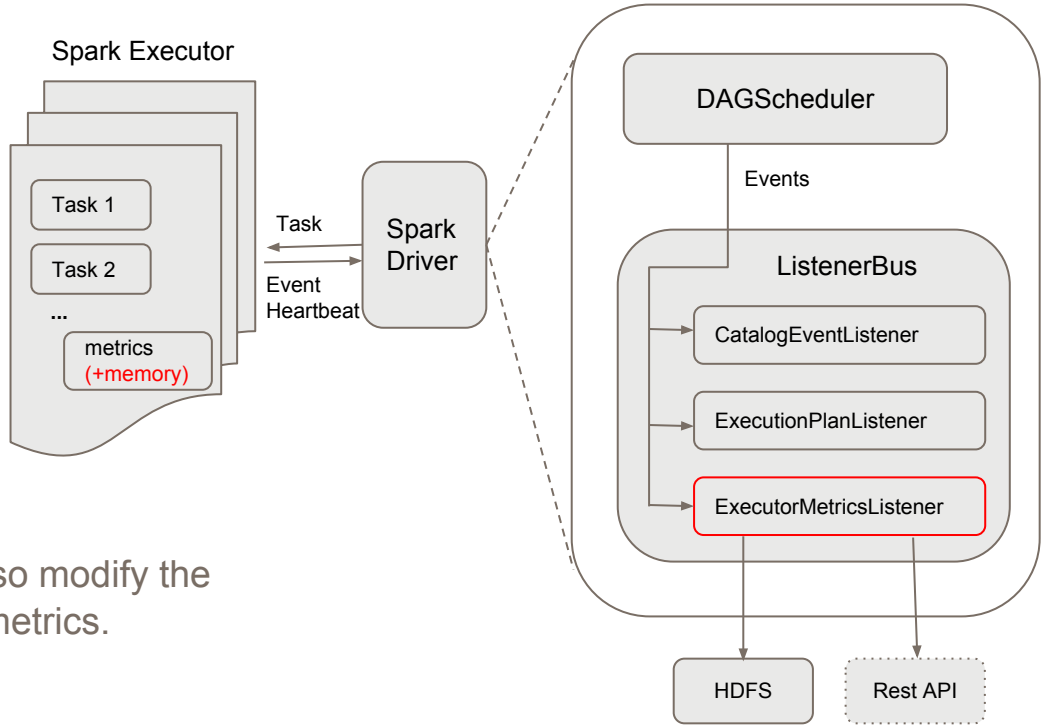


Job Processing



Profile listener

- Collect/dump extra metrics
  - Real memory usage
  - PRC count
  - Input/Output



* With this version spark profiler, we also modify the Spark Core to expose memory related metrics.

Success Cases

- Reduce High RPC Jobs
- Reduce Account Usage
- High Resource Failure jobs
- Retire Useless Jobs
- Historical based optimization
- Running job issue detection

Reduce High RPC Jobs

Mapper Rpc Call

CRITICAL

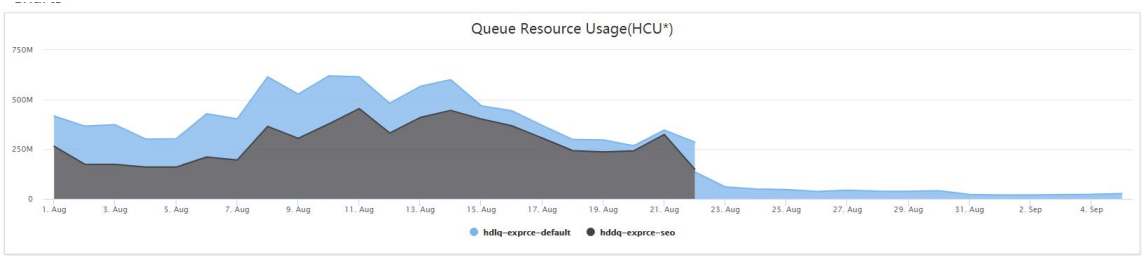
```
Number of tasks = 13962
Average task duration = 27 min 35 sec
Average RPC call number = 3207
Average RPC call/second = 12
```

Mapper Rpc Call

INFO

```
Number of tasks = 11162
Average task duration = 3 min 59 sec
Average RPC call number = 4
Average RPC call/second = 0
```

Reduce Account Usage



Job Running Time for all sites

Aug 1: 8:02 - 14:23
Sep 1: 8:02 - 11:29

High Resource Failure Jobs

areslvs | apollophx | artemislvs | herculeslvs

Search: spark-dsbe-pipeline | Queue Name: (All) | Owner: | Job Status: FAILED | Job Type: (All)

Absolute HCU(%): (All) | Absolute Waste HCU(%): (All) | Waste HCU(%): (All) | Suggestion Level: (All)

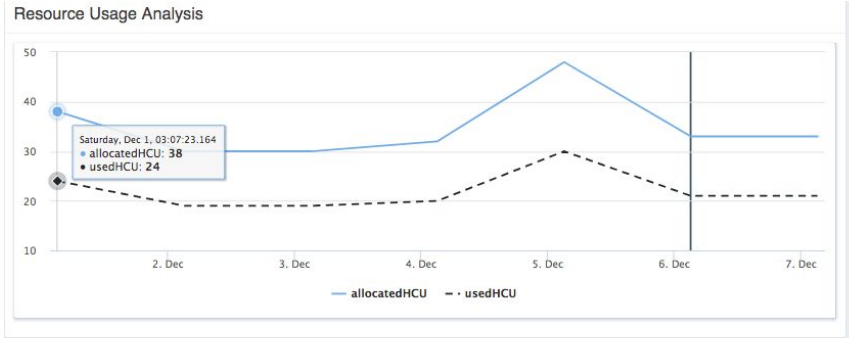
Query Jobs

Type	Job Id	Job Name	Suggestion Level	User	Queue	Allocated HCU(GB.s)	Wasted HCU(GB.s)	RPC	Start Time	End Time
SPARK	application_1531969122023_251241	spark-dsbe-pipeline	1 CRITICAL, 1 WARNING			2,918,917,545	0	-	2018-08-03 13:47:32	2018-08-06 00:07:59
SPARK	application_1528948548014_452526	spark-dsbe-pipeline	1 WARNING			2,523,956,356	0	-	2018-07-06 13:47:30	2018-07-08 02:07:51
SPARK	application_1535060340021_17006	spark-dsbe-pipeline	1 CRITICAL, 1 WARNING			1,948,112,135	0	0	2018-08-24 13:58:35	2018-08-25 20:31:24
SPARK	application_1531969122023_485341	spark-dsbe-pipeline	1 WARNING			1,856,933,348	0	-	2018-08-10 13:48:22	2018-08-12 03:00:43
SPARK	application_1531969122023_29602	spark-dsbe-pipeline	142 CRITICAL, 4 WARNING			1,393,662,208	0	-	2018-07-20 13:47:53	2018-07-22 05:02:53
SPARK	application_1531969122023_654303	spark-dsbe-pipeline	1 CRITICAL, 1 WARNING			1,368,726,653	0	-	2018-08-17 13:48:26	2018-08-18 11:58:35
SPARK	application_1531969122023_143525	spark-dsbe-pipeline	4 CRITICAL, 1 WARNING			107,115,294	0	-	2018-07-27 13:47:45	2018-07-29 11:20:40
SPARK	application_1528948548014_568201	spark-dsbe-pipeline	1 WARNING			9,841,817	0	-	2018-07-13 13:48:20	2018-07-14 17:53:21

Retire Useless Jobs

Table/Folder	Job Resource	Cluster Percentage
		Apollo (1.3%)
		Ares（0.4%）
		Ares (0.15%)

Historical based optimization



Job List 1 (7)

job_1542615348024_87224

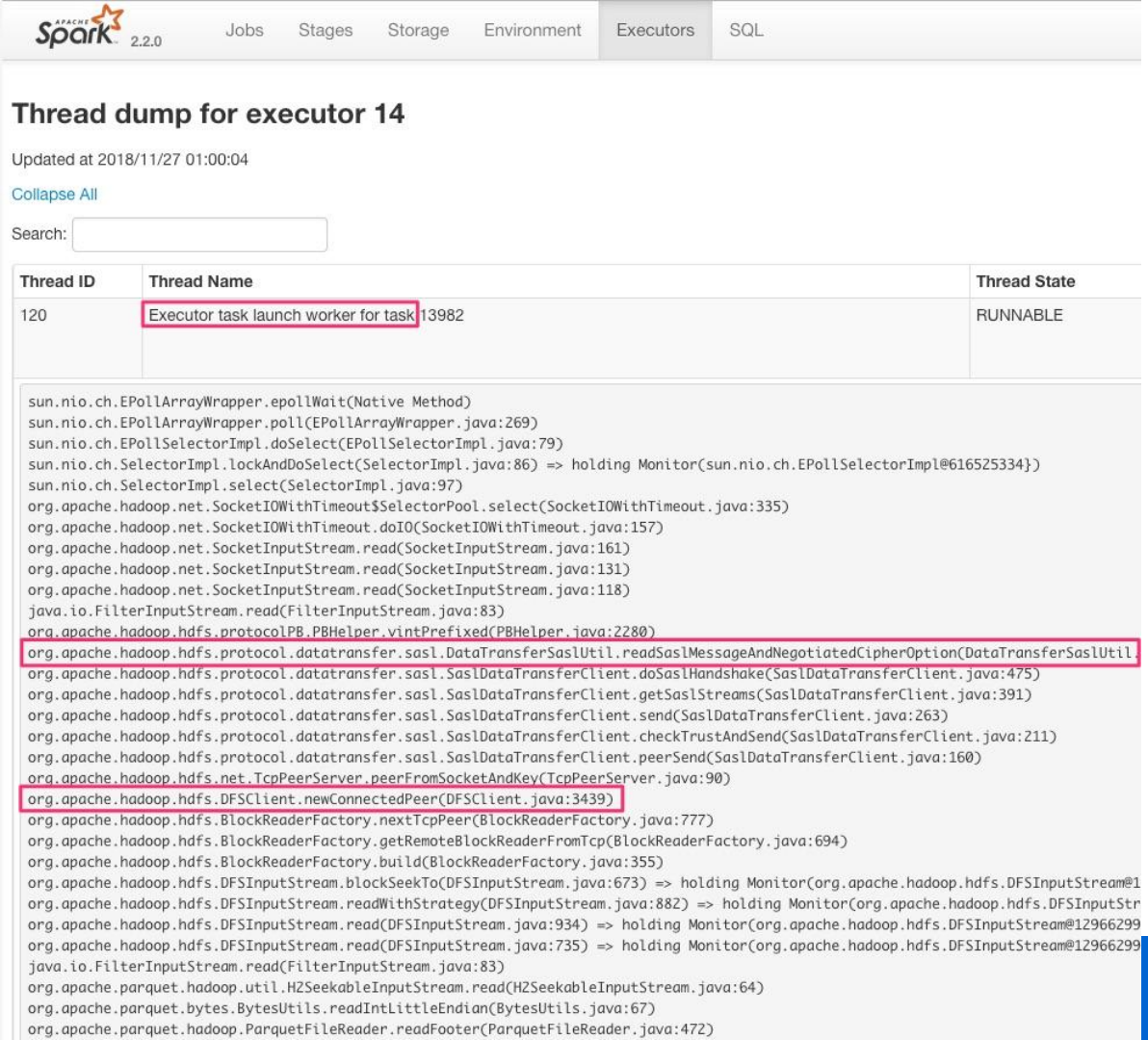
Final Status	SUCCEEDED
Launch Time	2018-12-01 03:07:23
Finish Time	2018-12-01 03:07:33
Wait Time	10s
Duration	18s
Total Launched Maps	1
Total Launched Reduces	0
Job Input Size	11837955
Allocated HCU(GB.s)	38
Max Used HCU(GB.s)	24

job_1542615348024_93795

Final Status	SUCCEEDED
Launch Time	2018-12-02 03:06:57
Finish Time	2018-12-02 03:07:06
Wait Time	9s
Duration	15s
Total Launched Maps	1
Total Launched Reduces	0
Job Input Size	11840892
Allocated HCU(GB.s)	30
Max Used HCU(GB.s)	19

Running job issue detection

- Current “Running job issue detection” could find out five kinds of cases:
  - Queue resource overload
  - Slow down by preemption
  - Shuffle Data skewing
  - Failure disk issue
  - Part of known Spark/Hadoop bug



Root cause is [HDFS-10223] peerFromSocketAndKey performs SASL exchange before setting connection timeouts

JPM need to detect this issue automatically. We first patch back [SPARK-23235]
Add executor Threaddump to api to our Spark.

Use JPM running job analysis to detect.

Issues may cause a hung task/job

https://issues.apache.org/jira/browse/SPARK-22172
https://issues.apache.org/jira/browse/SPARK-22074
https://issues.apache.org/jira/browse/SPARK-18971
https://issues.apache.org/jira/browse/SPARK-21928
https://issues.apache.org/jira/browse/SPARK-22083
https://issues.apache.org/jira/browse/SPARK-14958
https://issues.apache.org/jira/browse/SPARK-20079
https://issues.apache.org/jira/browse/SPARK-13931
https://issues.apache.org/jira/browse/SPARK-19617
https://issues.apache.org/jira/browse/SPARK-23365
https://issues.apache.org/jira/browse/SPARK-21834
https://issues.apache.org/jira/browse/SPARK-19631
https://issues.apache.org/jira/browse/SPARK-21656

MISC

- Self monitor and alert for latency, failure rate
- Low latency

  Rewrite json deserialize part, skip useless metrics
- High Success rate
  - Avoid OOM

  Persist entities ASAP to lower memory usage

  Metrics Based Suggestion Rule Engine listener: register needed entities/metrics and only maintain needed data.
  - Group failure reason to improve

JPM vs Dr. Elephant

Features which JPM could and Dr. elephant couldn't	
User bound	Since JPM is part of our one-stop management system, we have all user related info.
Diagnostics	JPM aggregates failed job logs, so user can easily get the failure trace for troubleshooting
Scalability	Dr elephant just launch a thread pool to do the job parsing which is hard to horizontally scale.
Maintenance	Per Dr elephant per cluster on one node, which is hard to manage and maintain.
Volume	Dr elephant uses mysql, if we want to save more data like task attempt entities in MR, it cannot fulfill our needs.
Veracity	Dr elephant uses sampling to avoid OOM, which sacrifices the veracity of result.
Availability	Dr elephant is single point failure system.
Variety	Analysis only based on metrics, no environment and cluster status information.
Relationship	Applications in Dr elephant has no relationship. Issues which caused by upstream can't be detected.
Histories	There is no historic relationship between different instances of same application.
Realtime	Jobs which only have been handled completely by SHS could be fetched by Dr elephant.
Runtime	Only finished job could be analyzed in Dr elephant.

DEMO

if we have time

Use case of underlying Elasticsearch

- Storage Design
- Object Relational Mapping
- Rest API Extension

Storage Chosen

elasticsearch

Storage consumption

- Data is about to consume 1.5TB disk space (2 replica)
- With data contraction/compress, single data copy in ES consume only 24% disk space comparing to Hbase in Task level
  - Without too complicated index included, ES performance well in storage area
- Since complicated index, ES consume almost 3 times disk space than Hbase in Job level
  - ES defaultly create index for almost all fields
  - When data structure is complex, we need to carefully handle index setup in ES


```
{
  "currentState": "SUCCEEDED",
  "startTime": 1509803440262,
  "endTime": 1509805465544,
  "duration": 0,
  "error": null,
  "jobCounters": {
    "MapTaskAttemptCounter": {
      "COMBINE_INPUT_RECORDS": 1900090,
      "COMBINE_OUTPUT_RECORDS": 1226052,
      "COMMITTED_HEAP_BYTES": 82888884224,
      "CPU_MILLISECONDS": 26889180,
      "FAILED_SHUFFLE": 0,
      "GC_TIME_MILLIS": 319635,
      "MAP_INPUT_RECORDS": 877368037,
      "MAP_OUTPUT_BYTES": 2113456402,
      "MAP_OUTPUT_MATERIALIZED_BYTES": 232421346,
      "MAP_OUTPUT_RECORDS": 1900090,
      "MERGED_MAP_OUTPUTS": 0,
      "PHYSICAL_MEMORY_BYTES": 73757208576,
      "SPILLED_RECORDS": 1226052,
      "SPLIT_RAW_BYTES": 13682,
      "TASK_ATTEMPT_DURATION": 25396206,
      "VIRTUAL_MEMORY_BYTES": 179590184960
    },
    "MapTaskAttemptFileSystemCounter": {
      "FILE_BYTES_READ": 0,
      "FILE_BYTES_WRITTEN": 237974516,
      "FILE_LARGE_READ_OPS": 0,
      "FILE_READ_OPS": 0,
      "FILE_WRITE_OPS": 0,
      "HDFS_BYTES_READ": 18847180713,
      "HDFS_BYTES_WRITTEN": 0,
      "HDFS_LARGE_READ_OPS": 0,
      "HDFS_READ_OPS": 380
    },
    "ReduceTaskAttemptCounter": {
      "COMBINE_INPUT_RECORDS": 568261,
      "COMBINE_OUTPUT_RECORDS": 146222,
      "COMMITTED_HEAP_BYTES": 8634499072,
      "CPU_MILLISECONDS": 120890,
      "FAILED_SHUFFLE": 0,
      "GC_TIME_MILLIS": 1630,
      "MERGED_MAP_OUTPUTS": 62,
```

Search performance

- Use official Java client to fetch data at hms016
- JobId exact match global matched (without time range specified)
  - In average it takes **264.6 ms** to fetch one Job fully infomration
- fuzzy search in jobName or jobId(without time range specified)
  -

Search key	Total result count	Time consume (ms, 1k lines)
	1476993	588
Information	1285	577
test	2993	551
select	469540	539
daily	8663	589

Search performance

- ‘in' action in job search

in' Search action	Total result count	Time consume (ms, 1k lines)
site in ["areslvs", "juno", "artemislvs"	4750065	608
queue in ["hddq-bpe", "hddq-gdi-ep", "hdmi-data", "hddq-gdi-kylin"]	632532	574
user in ["h 	295867	584

- Combination search in job search

Combination Search action	Total result count	Time consume (ms, 1k lines)
startTime in range[1514822400000, 1514908800000] and site in ["areslvs", "juno", "artemislvs"] and queue = "hddp-gdi-kylin" and currentState = "SUCCEEDED" limit 1000	485	435

Search performance

- Job statistic report

Statistic Search action	Total result count	Result	Time consume (ms)
18年1月2日所有job的map hdfs read总量			188
18年1月2日整天每个site的job map gc time时间总数 (group by site)			229

- Task statistic report

Statistic Search action	Total result count	Result	Time consume (ms)
jobid = "job_1513633548012_9024" hdfs read总量	64227	7.37 TB	196
jobid = "job_1513633548012_9464" task 运行的总量	5960	6383280s	183

- Concurrent Read

Concurrent Search action	Average Time Response (ms)
1000 threads concurrently read ES with random search condition, sleep for 500ms every round	1039

Write performance

- Bulk write with single thread
  - Bigger bulk size leads to better throughput, while, the largest bulk size ES supported is 10000.

Concurrent write action	Target	Thoughput (record/s)
10000 bulk size	mapreduce job	2572
	mapreduce job task	5655
1000 bulk size	mapreduce job	2083
	mapreduce job task	3460
500 bulk size	mapreduce job	1694
	mapreduce job task	2414

Write performance

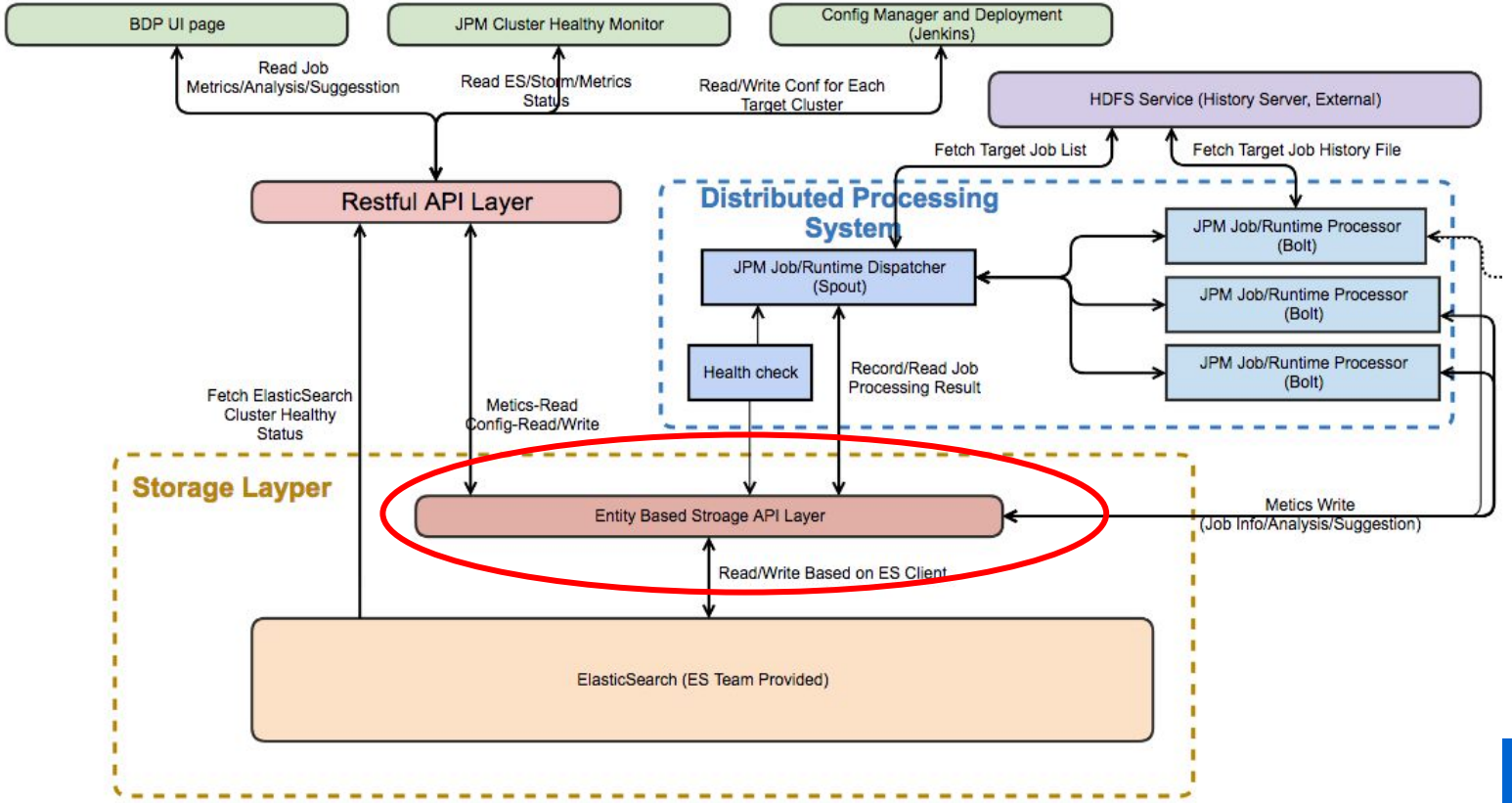
- Bulk write with multi threads
  - Each thread sleep about 30s each round, in such case, small bulk suffers.
  - Need large heap size, small one may lead to OOM.

Concurrent write action	Target	Thoughput (record/s)
100 threads 10000 bulk size	mapreduce job	14215
	mapreduce job task	25559
100 threads 1000 bulk size	mapreduce job	2198
	mapreduce job task	3036
100 threads 500 bulk size	mapreduce job	1213
	mapreduce job task	1222

Usage of underlying Elasticsearch


```
{
"cluster_name": "es6hadoop02",
"status": "green",
"timed_out": false,
"number_of_nodes": 11,
"number_of_data_nodes": 6,
"active_primary_shards": 6112,
"active_shards": 12224,
"relocating_shards": 0,
"initializing_shards": 0,
"unassigned_shards": 0,
"delayed_unassigned_shards": 0,
"number_of_pending_tasks": 0,
"number_of_in_flight_fetch": 0,
"task_max_waiting_in_queue_millis": 0,
"active_shards_percent_as_number": 100
}
```

ORM module: JPM-Valhalla



Elasticsearch ORM support

- Why we need an ORM of Elasticsearch?
  - Object-oriented friendly interface
  - Business decoupling
  - Have a chance to optimize in middleware
  - Easy to use

Annotation based ORM

- Annotation based Java entity to ES index mapping

```java
@Target(ElementType.TYPE)
@Retention(RetentionPolicy.RUNTIME)
public @interface ElasticSearchMeta {
    String index();

    String type() default "data";

    String[] id() default {};

    int shards() default 13;

    int replicas() default 1;

    boolean retire() default false;

    String view() default "";

    ElasticSearchRollingGranularity granularity() default ElasticSearchRollingGranularity.DAY;

    int limit() default 0;

    boolean fullindex() default false;
}
```

```java
@Target(ElementType.FIELD)
@Retention(RetentionPolicy.RUNTIME)
public @interface ElasticSearchIndex {
    ElasticSearchFieldType type() default ElasticSearchFieldType.AUTO;
    boolean enabled() default true;
}
```

Supported types

```
// elasticsearch types:
// https://www.elastic.co/guide/en/elasticsearch/reference/current/mapping-types.html

public enum ElasticSearchFieldType {
    TEXT,            // with word split
    KEYWORD,        // full text value
    LONG,
    INTEGER,
    SHORT,
    BYTE,
    FLOAT,
    DOUBLE,
    DATE,
    BOOLEAN,
    BINARY,
    ARRAY,
    OBJECT,
    NESTED,
    AUTO,           // auto detect, default value
    NONE            // unknown version
}
```

Field datatypes

Elasticsearch supports a number of different datatypes for the fields in a document:

Core datatypes

string

`text` and `keyword`

Numeric datatypes

`long`, `integer`, `short`, `byte`, `double`, `float`, `half_float`, `scaled_float`

Date datatype

`date`

Boolean datatype

`boolean`

Binary datatype

`binary`

Range datatypes

`integer_range`, `float_range`, `long_range`, `double_range`, `date_range`

Complex datatypes

Array datatype

Array support does not require a dedicated `type`

Object datatype

`object` for single JSON objects

Nested datatype

`nested` for arrays of JSON objects

Elasticsearch ORM support

```
@ElasticSearchMeta(index = "mr_task_entity",
    type = "data",
    id = {"site", "jobId", "taskId"},
    retire = true,
    granularity = ElasticSearchRollingGranularity.DAY,
    view = "mr_task_entity",
    limit = 30)
public class MRTaskEntity extends SiteBasedEntity {

    @ElasticSearchIndex
    private String jobId;

    @ElasticSearchIndex
    private String taskId;

    @ElasticSearchIndex
    private long startTime = 0L;
    @ElasticSearchIndex
    private long finishTime = 0L;
    private long launchTime = 0L; //first task attempt start time

    @ElasticSearchIndex
    private String taskType;

    @ElasticSearchIndex
    private String status;

    @ElasticSearchIndex
    private String error;
```

Supported operations

Index related:

```
public interface ElasticExecutor extends Closeable {

    // auto generate based on clz
    // if retired == 0 ignore date information
    // otherwise treat it as createIndex(clz, today)
    boolean createIndex(Class clz);

    // create with specified data information
    boolean createIndex(Class clz, Calendar day);

    // remove index
    boolean deleteIndex(Class clz);

    // remove with specified data information
    boolean deleteIndex(Class clz, Calendar day);

    boolean deleteIndex(String index);

    // test if index is existed
    boolean existedIndex(Class clz);

    // test if index is existed
    boolean existedIndex(Class clz, Calendar day);

    // test if index is existed
    boolean existedIndex(String index, String type);

    // count documents in index
    long countIndex(Class clz);

    // clear index data
    boolean clearIndex(Class clz);
```

Document related:

```
// insert with id
boolean insertById(Object object, String id);

boolean insertById(Object object, String id, Calendar

// insert with no id
boolean insertByNoId(Object object);

boolean insertByNoId(Object object, Calendar day);

// normal insert, id is judged by @ElasticSearchMeta
boolean insert(Object object);

boolean insert(Object object, Calendar day);

//delete by id
boolean delete(Class clz, String id);

//delete by id
boolean delete(Class clz, Calendar day, String id);
```

Aggregation related:

```
long count(Class clz);

long count(Class clz, ElasticFilter[] filter

long count(String index);

long count(String index, ElasticFilter[] fil

long count(String index, JsonObject query);
```

Alias related:

```
// create up alias
boolean addAlias(Class clz);

// create up alias with specified day
boolean addAlias(Class clz, Calendar day);

// remove up alias
boolean removeAlias(Class clz);

// remove up alias with specified day
boolean removeAlias(Class clz, Calendar day)

boolean removeAlias(String index, String ali

String[] allAliasIndex(Class clz);

boolean aliasesAction(ElasticAliasAction[] a
```

Complex read/write operations:

```
// bulkInsert multi values, values may not in same type, use today as day key
<T> boolean bulkInsert(T[] objects);

<T> boolean bulkInsert(List<T> objects);

// bulkInsert multi values, values may not in same type
<T> boolean bulkInsert(T[] objects, Calendar[] days);

<T> boolean bulkInsert(List<T> objects, List<Calendar> days);

<T extends Object> T[] search(Class<T[]> arrayClz, List<ElasticFilter> filters);

<T extends Object> T[] search(Class<T[]> arrayClz, ElasticFilter[] filters);

<T extends Object> T[] search(Class<T[]> arrayClz, List<ElasticFilter> filters, int size);

<T extends Object> T[] search(Class<T[]> arrayClz, ElasticFilter[] filters, int size);

<T extends Object> T[] search(Class<T[]> arrayClz, List<ElasticFilter> filters, int size, List<ElasticSort> sorts);

<T extends Object> T[] search(Class<T[]> arrayClz, ElasticFilter[] filters, int size, List<ElasticSort> sorts);

JsonArray rawSearch(Class clz, ElasticFilter[] filters, int size, String[] includeFields, ElasticSort[] sorts);
JsonArray rawSearch(String index, String type, ElasticFilter[] filters, int size, String[] includeFields, ElasticSort[] sorts);

<T extends Object> T[] scroll(Class<T[]> arrayClz, List<ElasticFilter> filters, int size, List<ElasticSort> sorts, int keepMin);

<T extends Object> T[] scroll(Class<T[]> arrayClz, ElasticFilter[] filters, int size, List<ElasticSort> sorts, int keepMin);

JsonArray rawScroll(Class arrayClz, ElasticFilter[] filters, int batchSize, String[] includeFields, ElasticSort[] sorts, int keepMin);

JsonArray rawScroll(String index, String type, ElasticFilter[] filters, int batchSize, String[] includeFields, ElasticSort[] sorts, int keepMin);
```

Advanced aggregation/join operations:

```
<T extends Object> Object aggregate(Class<T[]> arrayClz,
                                    List<ElasticFilter> filters,
                                    ElasticAggregate[] aggregates);

<T extends Object> Object aggregate(Class<T[]> arrayClz,
                                    ElasticFilter[] filters,
                                    ElasticAggregate[] aggregates);

JsonElement rawAggregate(Class clz, ElasticFilter[] filters, ElasticAggregate[] aggregates, String[] includeFields);
JsonElement rawAggregate(String index, String type, ElasticFilter[] filters, ElasticAggregate[] aggregates, String[] includeFields);

// special request
JsonObject raw(String method, String path, Map<String, String> params, String query);

JsonArray rawInnerJoin(String index1, ElasticFilter[] filters1,
                       String index2, ElasticFilter[] filters2,
                       String onKey, String[] includeFields, int maxSize);

// @NOTE temp solution for job handler
JsonArray rawInnerJoin(String index1, JsonObject queryObj1,
                      String index2, ElasticFilter[] filters2,
                      String onKey1, String onKey2, String[] includeFields, int maxSize);

long rawInnerJoinOnlyCount(String index1, JsonObject queryObj1,
                           String index2, ElasticFilter[] filters2,
                           String onKey1, String onKey2, String[] includeFields);
```

Join Example

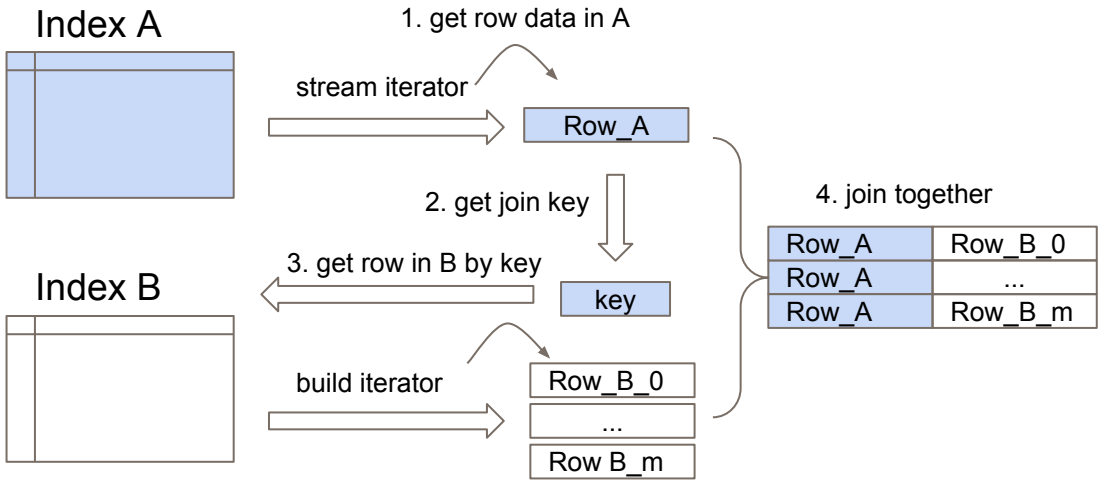
```
JsonArray result = executor.rawInnerJoin("mr_job_entity_2018", new ElasticFilter[]{
   new ElasticTermFilter("site", "areslvs"),
   new ElasticTermFilter("jobType", "MAPREDUCE")
}, "analysis_result_mr_2018", new ElasticFilter[]{
   new ElasticTermFilter("site", "areslvs"),
   new ElasticTermFilter("severity", "CRITICAL")
}, "jobId", new String[]{"jobId","jobName", "jobType", "userName", "queueName", "jobNormalizedName"}, 10000);

System.out.println(result.toString());
```

```java
// do multi round in case overflow
// 1. check total matched count in each table
// 2. if one is small (<1000), use it to directly join another
// 3. if both large, 1000 each to do join
// @TODO may include multi-thread to improve performance
@Override
public JsonArray rawInnerJoin(String index1, ElasticFilter[] filters1, String index2,
                              ElasticFilter[] filters2, String onKey,
                              String[] includeFields, int maxSize) {
    // check maxSize which should between 1-10000
    if (maxSize < MIN_OPS_SIZE) {
        maxSize = MIN_OPS_SIZE;
    } else if (maxSize > MAX_OPS_SIZE) {
        maxSize = MAX_OPS_SIZE;
    }

    // check total matched count
    long count1 = this.count(index1, filters1);
    long count2 = this.count(index2, filters2);

    JsonArray outputResult = null;

    // if one of any is small (<1000),  directly shuffle the small one and join them
    if (count1 < BROADCAST_THREHOLD) {
        outputResult = rawBroadcastInnerJoin(index1, filters1, index2, filters2, onKey, includeFields);
    } else if (count2 < BROADCAST_THREHOLD) {
        outputResult = rawBroadcastInnerJoin(index2, filters2, index1, filters1, onKey, includeFields);
    } else if (count1 <= count2) {
        // pick the small one as fragment shuffle base
        outputResult = rawSplitedBroadcastInnerJoin(index1, filters1, count1,
            index2, filters2, count2,
            onKey, includeFields, maxSize);
    } else {
        outputResult = rawSplitedBroadcastInnerJoin(index2, filters2, count2,
            index1, filters1, count1,
            onKey, includeFields, maxSize);
    }

    if (outputResult != null) {
        return cutoffResult(outputResult, maxSize);
    } else {
        return null;
    }
}
```

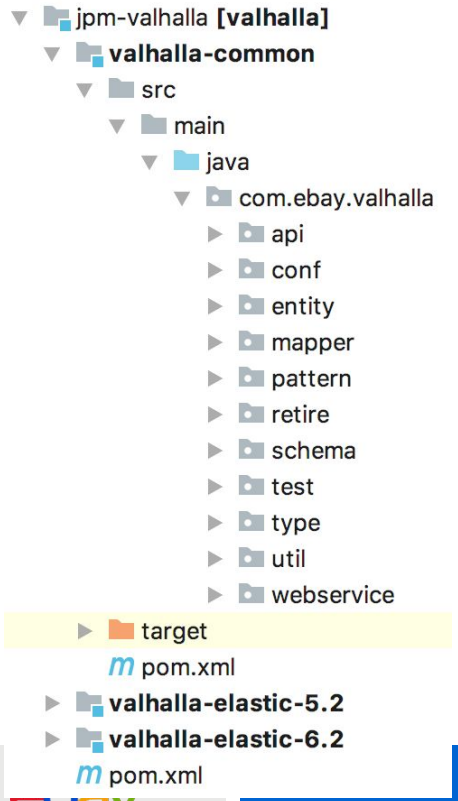
```java
//@TODO first just support join on string, late support on int/long/double
//      code for them all similar but need one better way to distinguish
public JsonArray rawBroadcastInnerJoin(String shuffleIndex, ElasticFilter[] shuffleFilters,
                                       String fullIndex, ElasticFilter[] fullFilters,
                                       String onKey, String[] includeFields) {

    // first check if the includeFields contains the onKey, we need it to do join
    // simple solution is to add it into includeFields directly
    String[] includeFieldsWithOnKey = Arrays.copyOf(includeFields, newLength: includeFields.length + 1);
    includeFieldsWithOnKey[includeFieldsWithOnKey.length - 1] = onKey;

    // fetch onKey in shuffleIndex
    JsonArray shuffleResults = this.rawSearch(shuffleIndex, type: "data", shuffleFilters,
        size: 0, includeFieldsWithOnKey, new ElasticSort[]{
            new ElasticFieldSort(onKey, ElasticSortOrder.asc)
        });

    // fetch one to check type
    JsonElement testOne = shuffleResults.get(0).getAsJsonObject().get(onKey);
    TermValueType valueType = JsonUtil.getJoinTermValueType(testOne);

    if (valueType == TermValueType.INVALID) {
        log.error("Unable to join in one Non-Primitive value key {}", onKey);
        // @TODO should throw error here..
        return null;
    }

    return broadcastInnerArrayJoin(shuffleResults, fullIndex, fullFilters, onKey, onKey,
        includeFields, includeFieldsWithOnKey, valueType);
}
```

```
public JsonArray broadcastInnerArrayJoin(JsonArray shuffleResults, String fullIndex, ElasticFilter[] fullFilters,
                                         String onKey1, String onKey2, String[] includeFields, String[] includeFieldsWithOnKey,
                                         TermValueType valueType) {
    // get all target on key
    Object[] keys = new String[shuffleResults.size()];

    // read key values
    for (int i = 0; i < keys.length; i++) {
        JsonObject jobj = shuffleResults.get(i).getAsJsonObject();
        JsonElement keyEle = jobj.get(onKey1);
        if (keyEle == null) {
            keyEle = jobj.get(onKey2);
        }
        keys[i] = JsonUtil.getPrimitiveValue(keyEle.getAsJsonPrimitive());
    }

    // add extra filter into full index search
    ElasticTermsFilter termsFilter = new ElasticTermsFilter(onKey2, keys, valueType);
    ElasticFilter[] fullFiltersWithKeys = Arrays.copyOf(fullFilters,  newLength: fullFilters.length + 1);
    fullFiltersWithKeys[fullFiltersWithKeys.length - 1] = termsFilter;

    JsonArray fullResults = this.rawSearch(fullIndex,  type: "data",
        fullFiltersWithKeys,  size: 10000, includeFieldsWithOnKey, new ElasticSort[]{
            new ElasticFieldSort(onKey2, ElasticSortOrder.asc)}
    );

    // now just join them all with sorted by onKey already
    return JsonUtil.innerJoinStringKeyArraysOnKey(shuffleResults, fullResults, onKey1, onKey2, includeFields);
}
```

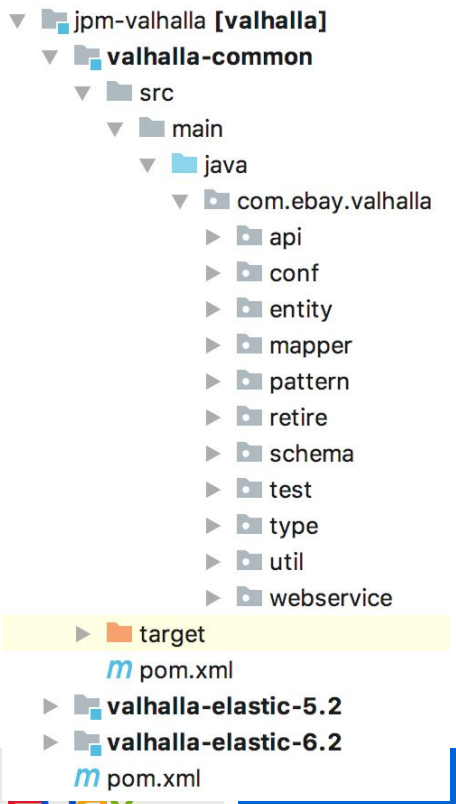
```java
// should be sorted based on onKey
// array1 key should be unique
// @TODO however only return one key for each
public static JsonArray innerJoinStringKeyArraysOnKey(JsonArray array1, JsonArray array2,
                                                      String onKey1, String onKey2, String[] includeFields) {
    JsonArray resultArray = new JsonArray();

    // if one of them are empty directly return empty value
    if (array1 == null || array1.size() == 0 ||
        array2 == null || array2.size() == 0) {
        return resultArray;
    }

    Iterator<JsonElement> iterator1 = array1.iterator();
    Iterator<JsonElement> iterator2 = array2.iterator();

    String latestKey1 = null;
    String latestKey2 = null;

    JsonObject currentEle1 = null;
    JsonObject currentEle2 = null;
    while (true) {
        // if no more in iterator1, break
        if (latestKey1 == null && !iterator1.hasNext()) {
            break;
        }

        // try to fetch one if current one is consumed in last round
        if (latestKey1 == null) {
            currentEle1 = iterator1.next().getAsJsonObject();
            if (currentEle1.has(onKey1)) {
                latestKey1 = currentEle1.get(onKey1).getAsString();
            } else {
                latestKey1 = currentEle1.get(onKey2).getAsString();
            }
        }

        // if no more in iterator1, break
        if (latestKey2 == null && !iterator2.hasNext()) {
            break;
        }

        // try to fetch one if current one is consumed in last round
        if (latestKey2 == null) {
            currentEle2 = iterator2.next().getAsJsonObject();
            latestKey2 = currentEle2.get(onKey2).getAsString();
        }

        int compareValue = latestKey1.compareTo(latestKey2);


        if (compareValue == 0) {
            // equal, consume both and put into result array
            resultArray.add(buildJsonObj(includeFields, currentEle1, currentEle2));
            latestKey1 = null;
            latestKey2 = null;
        } else if (compareValue > 0) {
            // key1 > key2, consume key2, keep key1
            latestKey2 = null;
        } else {
            // key1 < key2, comsume key1, keep key2
            latestKey1 = null;
        }
    }

    return resultArray;
}
```

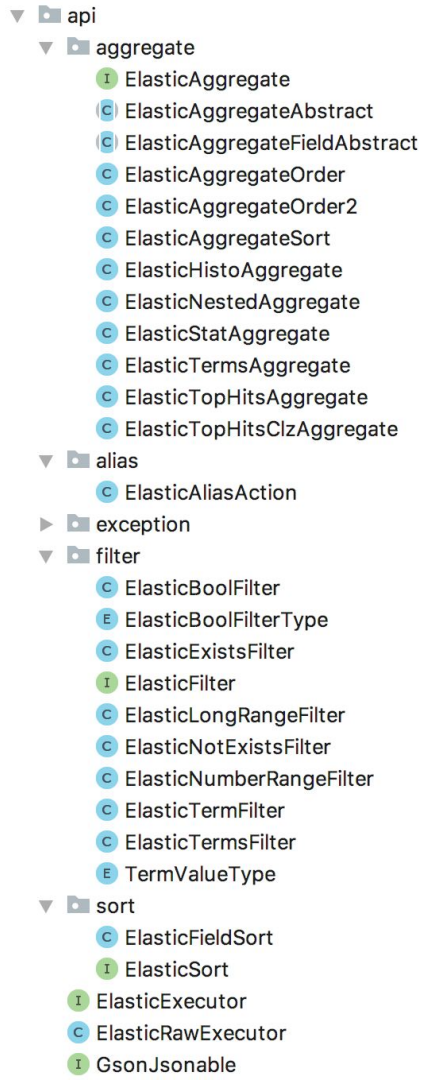


Source Code

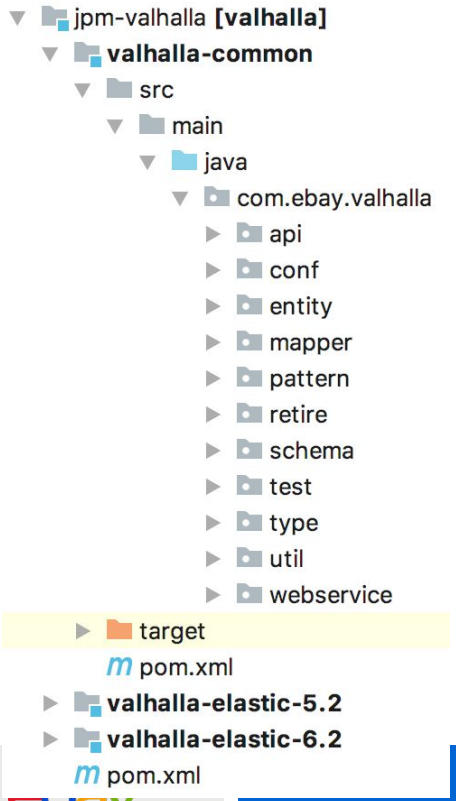

jpm-valhalla [valhalla]
valhalla-common
src
main
java
com.ebay.valhalla
api
conf
entity
mapper
pattern
retire
schema
test
type
util
webservice
target
pom.xml
valhalla-elastic-5.2
valhalla-elastic-6.2
pom.xml

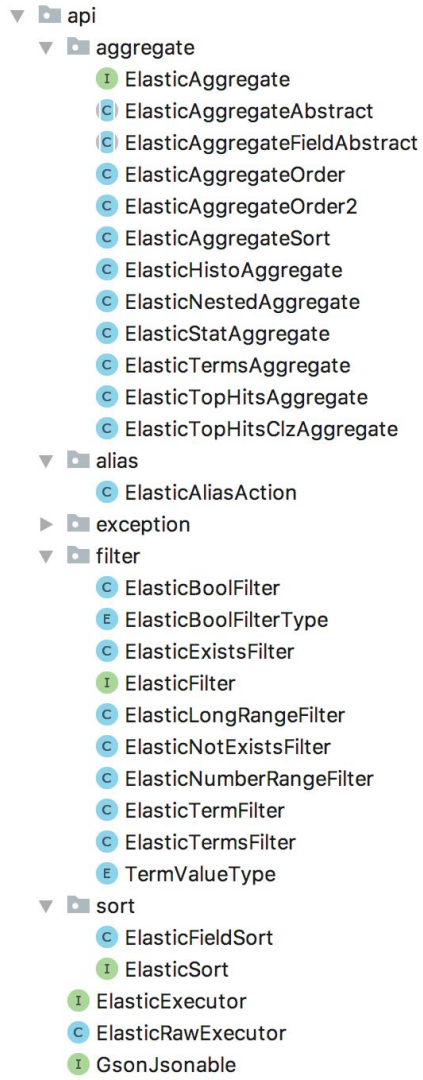
Source Code

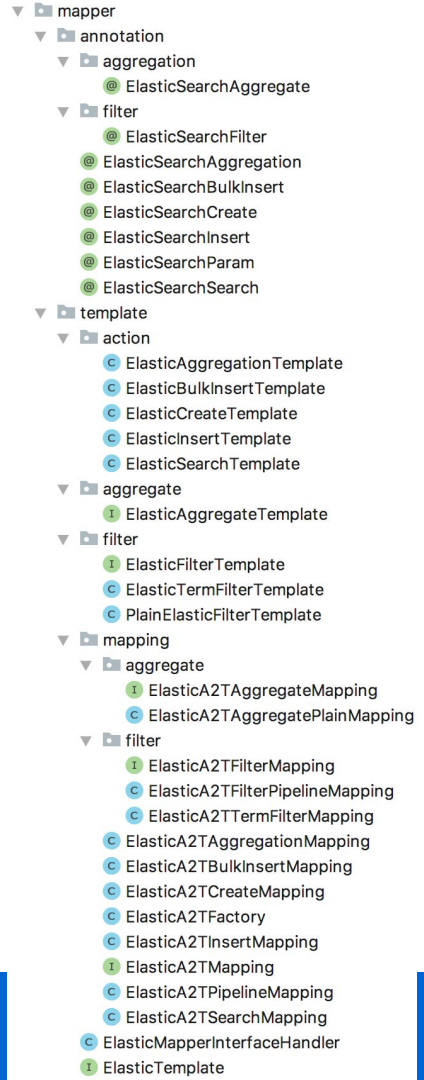

jpm-valhalla [valhalla]
valhalla-common
src
main
java
com.ebay.valhalla
api
conf
entity
mapper
pattern
retire
schema
test
type
util
webservice
target
pom.xml
valhalla-elastic-5.2
valhalla-elastic-6.2
pom.xml


api
aggregate
ElasticAggregate
ElasticAggregateAbstract
ElasticAggregateFieldAbstract
ElasticAggregateOrder
ElasticAggregateOrder2
ElasticAggregateSort
ElasticHistoAggregate
ElasticNestedAggregate
ElasticStatAggregate
ElasticTermsAggregate
ElasticTopHitsAggregate
ElasticTopHitsClzAggregate
alias
ElasticAliasAction
exception
filter
ElasticBoolFilter
ElasticBoolFilterType
ElasticExistsFilter
ElasticFilter
ElasticLongRangeFilter
ElasticNotExistsFilter
ElasticNumberRangeFilter
ElasticTermFilter
ElasticTermsFilter
TermValueType
sort
ElasticFieldSort
ElasticSort
ElasticExecutor
ElasticRawExecutor
GsonJsonable

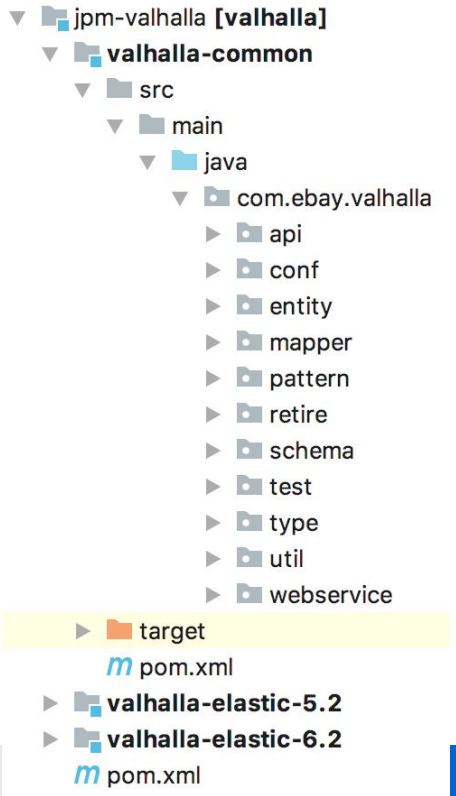
Source Code

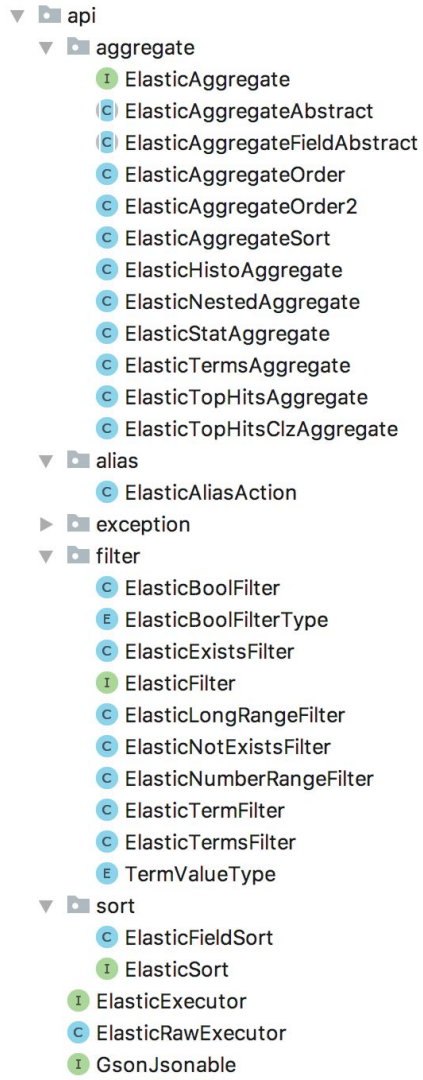

jpm-valhalla [valhalla]
valhalla-common
src
main
java
com.ebay.valhalla
api
conf
entity
mapper
pattern
retire
schema
test
type
util
webservice
target
pom.xml
valhalla-elastic-5.2
valhalla-elastic-6.2
pom.xml

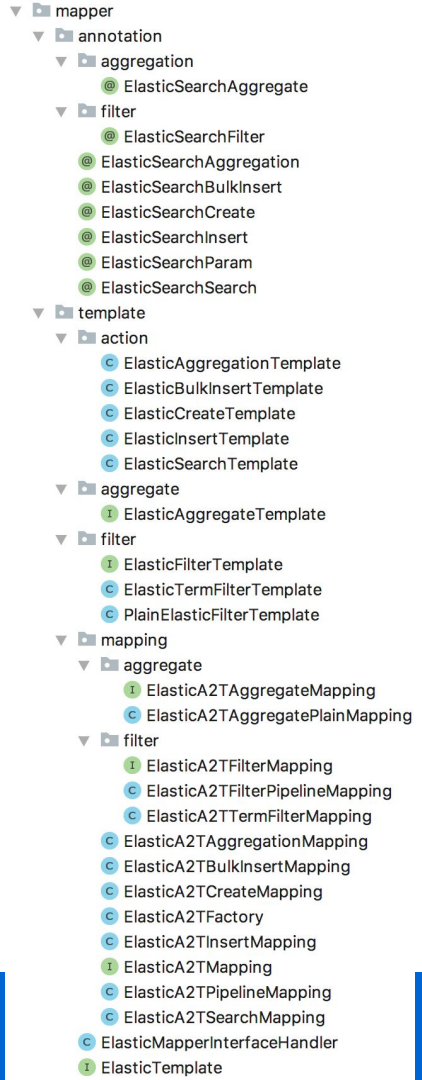

api
aggregate
ElasticAggregate
ElasticAggregateAbstract
ElasticAggregateFieldAbstract
ElasticAggregateOrder
ElasticAggregateOrder2
ElasticAggregateSort
ElasticHistoAggregate
ElasticNestedAggregate
ElasticStatAggregate
ElasticTermsAggregate
ElasticTopHitsAggregate
ElasticTopHitsClzAggregate
alias
ElasticAliasAction
exception
filter
ElasticBoolFilter
ElasticBoolFilterType
ElasticExistsFilter
ElasticFilter
ElasticLongRangeFilter
ElasticNotExistsFilter
ElasticNumberRangeFilter
ElasticTermFilter
ElasticTermsFilter
TermValueType
sort
ElasticFieldSort
ElasticSort
ElasticExecutor
ElasticRawExecutor
GsonJsonable

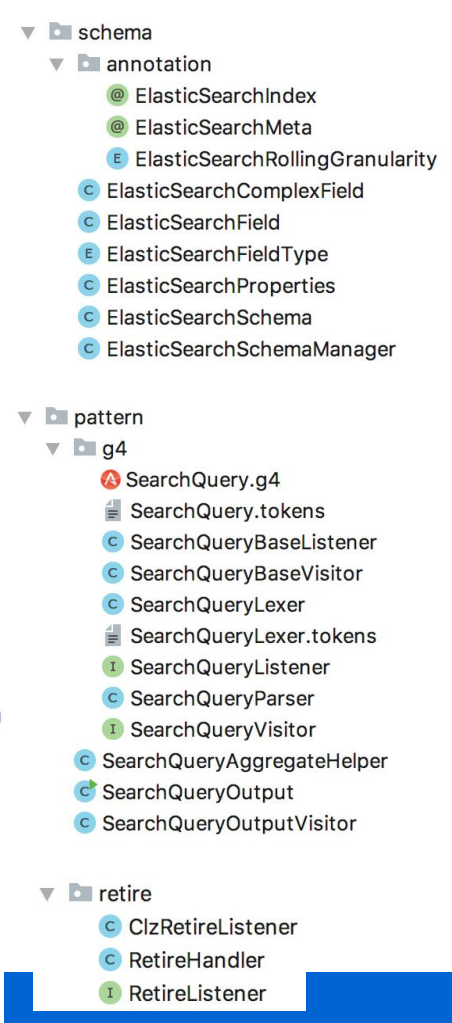

mapper
annotation
aggregation
ElasticSearchAggregate
filter
ElasticSearchFilter
ElasticSearchAggregation
ElasticSearchBulkInsert
ElasticSearchCreate
ElasticSearchInsert
ElasticSearchParam
ElasticSearchSearch
template
action
ElasticAggregationTemplate
ElasticBulkInsertTemplate
ElasticCreateTemplate
ElasticInsertTemplate
ElasticSearchTemplate
aggregate
ElasticAggregateTemplate
filter
ElasticFilterTemplate
ElasticTermFilterTemplate
PlainElasticFilterTemplate
mapping
aggregate
ElasticA2TAggregateMapping
ElasticA2TAggregatePlainMapping
filter
ElasticA2TFilterMapping
ElasticA2TFilterPipelineMapping
ElasticA2TTermFilterMapping
ElasticA2TAggregationMapping
ElasticA2TBulkInsertMapping
ElasticA2TCreateMapping
ElasticA2TFactory
ElasticA2TInsertMapping
ElasticA2TMapping
ElasticA2TPipelineMapping
ElasticA2TSearchMapping
ElasticMapperInterfaceHandler
ElasticTemplate

Source Code

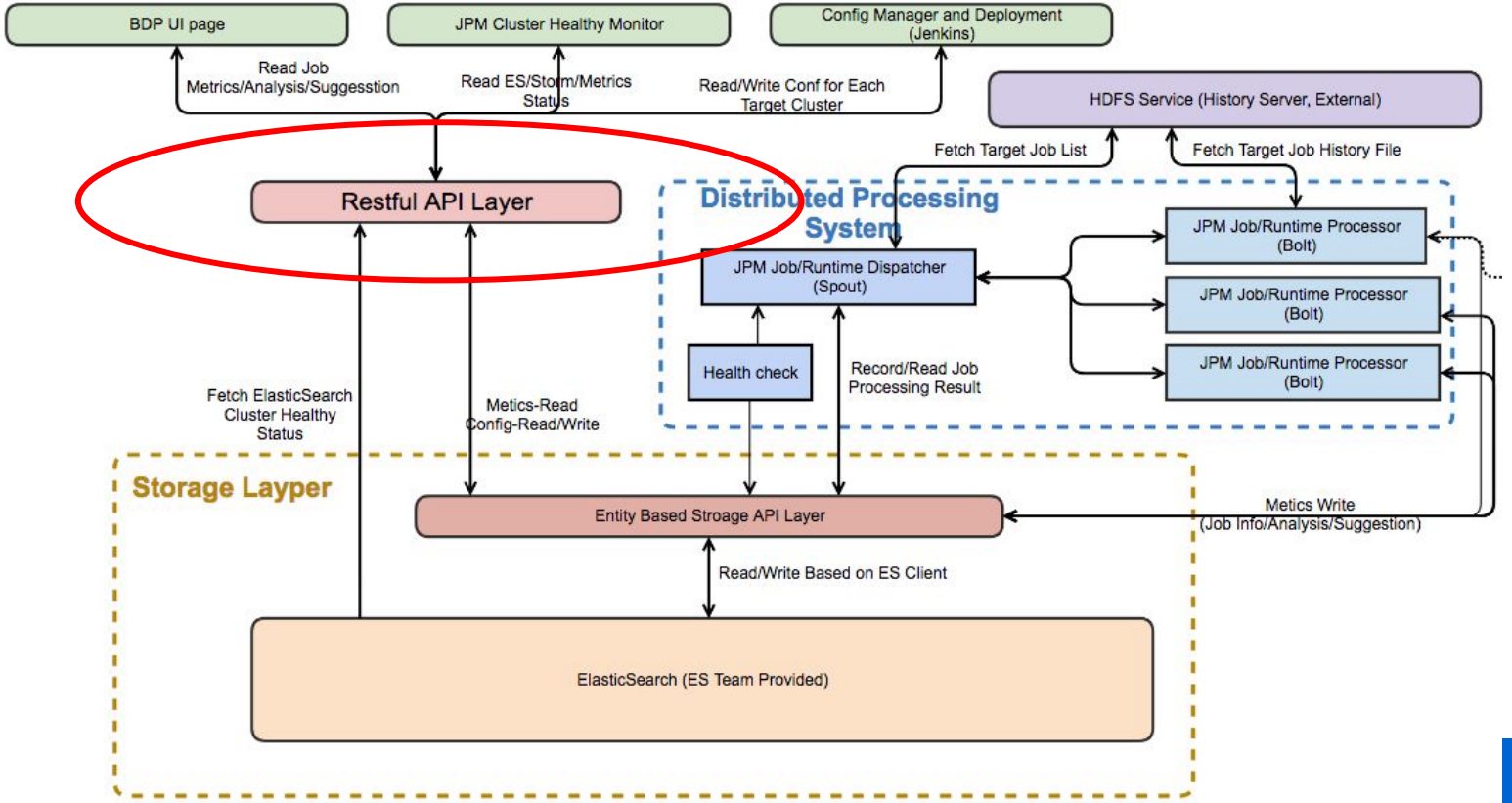

jpm-valhalla [valhalla]
valhalla-common
src
main
java
com.ebay.valhalla
api
conf
entity
mapper
pattern
retire
schema
test
type
util
webservice
target
pom.xml
valhalla-elastic-5.2
valhalla-elastic-6.2
pom.xml


api
aggregate
ElasticAggregate
ElasticAggregateAbstract
ElasticAggregateFieldAbstract
ElasticAggregateOrder
ElasticAggregateOrder2
ElasticAggregateSort
ElasticHistoAggregate
ElasticNestedAggregate
ElasticStatAggregate
ElasticTermsAggregate
ElasticTopHitsAggregate
ElasticTopHitsClzAggregate
alias
ElasticAliasAction
exception
filter
ElasticBoolFilter
ElasticBoolFilterType
ElasticExistsFilter
ElasticFilter
ElasticLongRangeFilter
ElasticNotExistsFilter
ElasticNumberRangeFilter
ElasticTermFilter
ElasticTermsFilter
TermValueType
sort
ElasticFieldSort
ElasticSort
ElasticExecutor
ElasticRawExecutor
GsonJsonable


mapper
annotation
aggregation
ElasticSearchAggregate
filter
ElasticSearchFilter
ElasticSearchAggregation
ElasticSearchBulkInsert
ElasticSearchCreate
ElasticSearchInsert
ElasticSearchParam
ElasticSearchSearch
template
action
ElasticAggregationTemplate
ElasticBulkInsertTemplate
ElasticCreateTemplate
ElasticInsertTemplate
ElasticSearchTemplate
aggregate
ElasticAggregateTemplate
filter
ElasticFilterTemplate
ElasticTermFilterTemplate
PlainElasticFilterTemplate
mapping
aggregate
ElasticA2TAggregateMapping
ElasticA2TAggregatePlainMapping
filter
ElasticA2TFilterMapping
ElasticA2TFilterPipelineMapping
ElasticA2TTermFilterMapping
ElasticA2TAggregationMapping
ElasticA2TBulkInsertMapping
ElasticA2TCreateMapping
ElasticA2TFactory
ElasticA2TInsertMapping
ElasticA2TMapping
ElasticA2TPipelineMapping
ElasticA2TSearchMapping
ElasticMapperInterfaceHandler
ElasticTemplate


schema
annotation
ElasticSearchIndex
ElasticSearchMeta
ElasticSearchRollingGranularity
ElasticSearchComplexField
ElasticSearchField
ElasticSearchFieldType
ElasticSearchProperties
ElasticSearchSchema
ElasticSearchSchemaManager
pattern
g4
SearchQuery.g4
SearchQuery.tokens
SearchQueryBaseListener
SearchQueryBaseVisitor
SearchQueryLexer
SearchQueryLexer.tokens
SearchQueryListener
SearchQueryParser
SearchQueryVisitor
SearchQueryAggregateHelper
SearchQueryOutput
SearchQueryOutputVisitor
retire
ClzRetireListener
RetireHandler
RetireListener

Restful API Extension



Restful API Extension

- Why we need an Restful API extension?
  - Use restful-API for frontend developer, more friendly
  - Lowest learning cost, avoiding bad usage, more efficiency
  - Hide the underlying implementation details, more safety
  - Pluggable authentication/authorization, more resilient

Elasticsearch Restful API Extension

- Based on Antlr4

Partial of SearchQuery.g4

```
grammar SearchQuery;
//[@site="[SITEPARAM]"]{@jobId,@jobDefId,@jobName,@currentState,@user,@queue,@startTime,@endTime,@jobType}&pageSize=[PAGESIZEPARAM]&start
Time=[STARTTIMEPARAM]&endTime=[ENDTIMEPARAM]
//JobProcessTimeStampService[@site="SITEPARAM"]<@site>{max(currentTimeStamp)}

query : clzName (filter)? (sort)? (selector)? (aggregator)? (search_max_size)?;
//query : value;

clzName : KEY;
filter : '['filter_list']' | '[]';
filter_list : (filter_item','filter_list) | filter_item;
filter_item : filter_item_equal | filter_item_range | filter_item_time_range | filter_item_compare | filter_item_terms;

selector : '{'selector_list'}' | '{}';
selector_list : (selector_item','selector_list) | selector_item;
selector_item : '@'KEY;

aggregator : '<'aggregator_list'>' | '<>';
aggregator_list : (aggregator_item','aggregator_list) | aggregator_item;
aggregator_item : aggregator_term_item | aggregator_top_item | aggregator_stat_item | aggregator_stat_list |aggregator_nested_item | aggregator_histo_item;
```

Example

api/elastic/search?query=spark_app_entity[@site="apollophx"]&@hcu<&size=100

```
spark_app_entity/_search
{
   “query”:{
      “term”:{
          “site”: {
            “value”:”apollophx”
          }
      }
   },
   "sort": [
     {
       "hcu": {
         "order": "desc"
       }
     }
    ],
   “size”:100
}
```

MISC

- Dual port restful executor
- RetireListener
- Gson Scala support
- ElasticSearchSchema.setFullCreationMapping(false)

Q & A

Thank you!

We are hiring!!
(Java/bigdata/payments/…)

专业、垂直、纯粹的 Elastic 开源技术交流社区

https://elasticsearch.cn/