Transwarp Hippo 1.0 用户手册

星环信息科技（上海）股份有限公司

2023-04

目录

图表目录

免责声明

本说明书依据现有信息制作,其内容如有更改,恕不另行通知。星环信息科技（上海）股份有限公司在编写该说 明书的时候已尽最大努力保证其内容准确可靠,但星环信息科技（上海）股份有限公司不对本说明书中的遗漏、 不准确或印刷错误导致的损失和损害承担责任。具体产品使用请以实际使用为准。

注释: Hadoop® 和 SPARK® 是 ApacheTM 软件基金会在美国和其他国家的商标或注册的商标。 Java® 是 Oracle 公司在美国和其他国家的商标或注册的商标。 Intel® 和 Xeon® 是英特尔公司在美国、中国和其他国 家的商标或注册的商标。

手册版本信息
版本号: T0019x1-95-010
发布日期: 2023-04

1. 产品简介

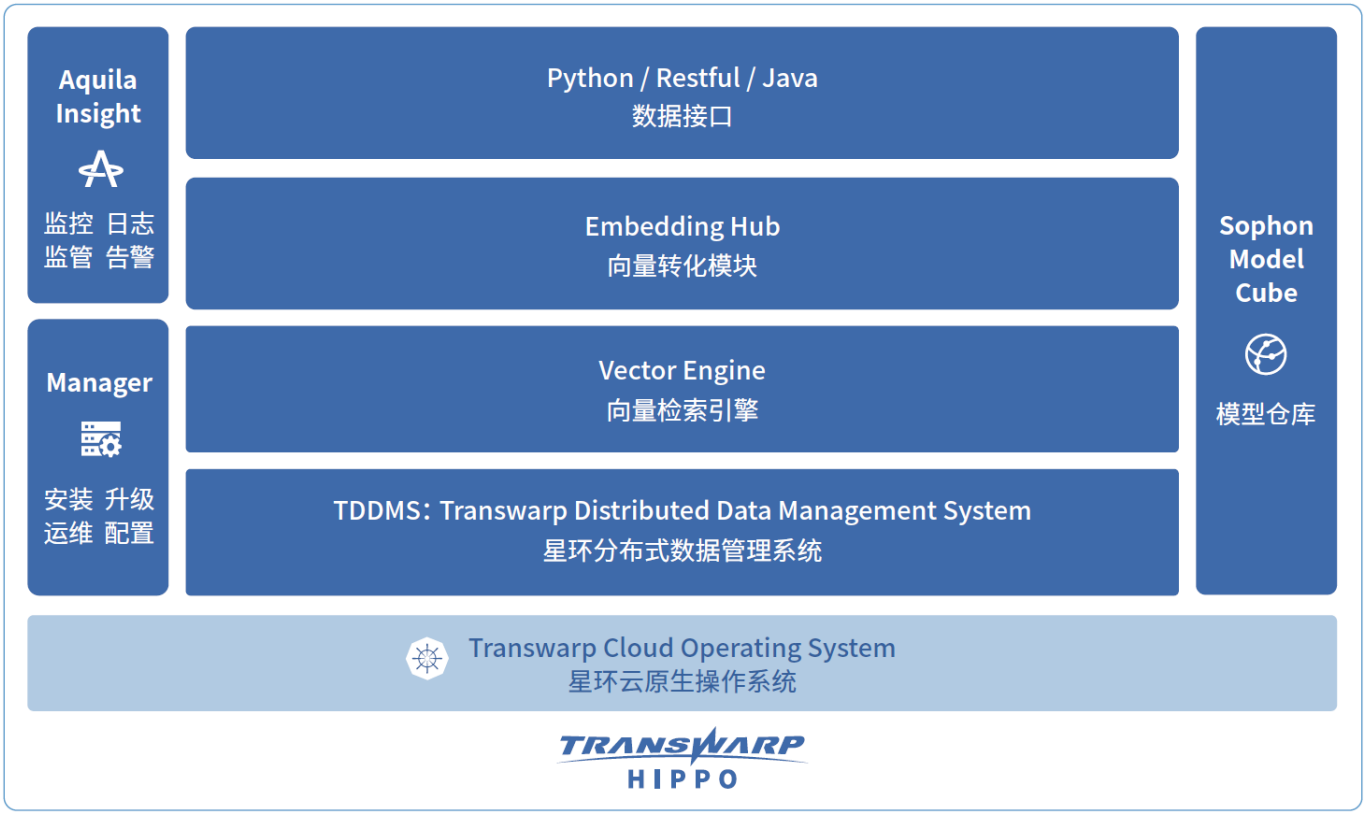
1.1. 产品定位

伴随着企业对海量非结构化数据管理的需求的不断加深，以及深度学习在工业界的广泛落地，向量数据在实际应用场景下的数据量级开始直线增加。想要高效处理这些海量的向量数据，就需要更细分、更专业的数据基础设施，为向量构建专门的数据库处理系统。

Transwarp Hippo (以下简称 Hippo) 是星环科技自主可控的一款企业级云原生分布式向量数据库，支持存储，索引以及管理海量的向量式数据集，能够高效的解决向量相似度检索以及高密度向量聚类等问题。Hippo 具备高可用，高性能，易拓展等特点，支持多种向量搜索索引，支持数据分区分片，数据持久化，增量数据摄取，向量标量字段过滤混合查询等功能，能够很好的满足企业针对海量向量数据的高实时性查询、检索、召回等场景。

1.2. 软件架构

1.2.1. 架构图



图表 1 Hippo 总体架构图

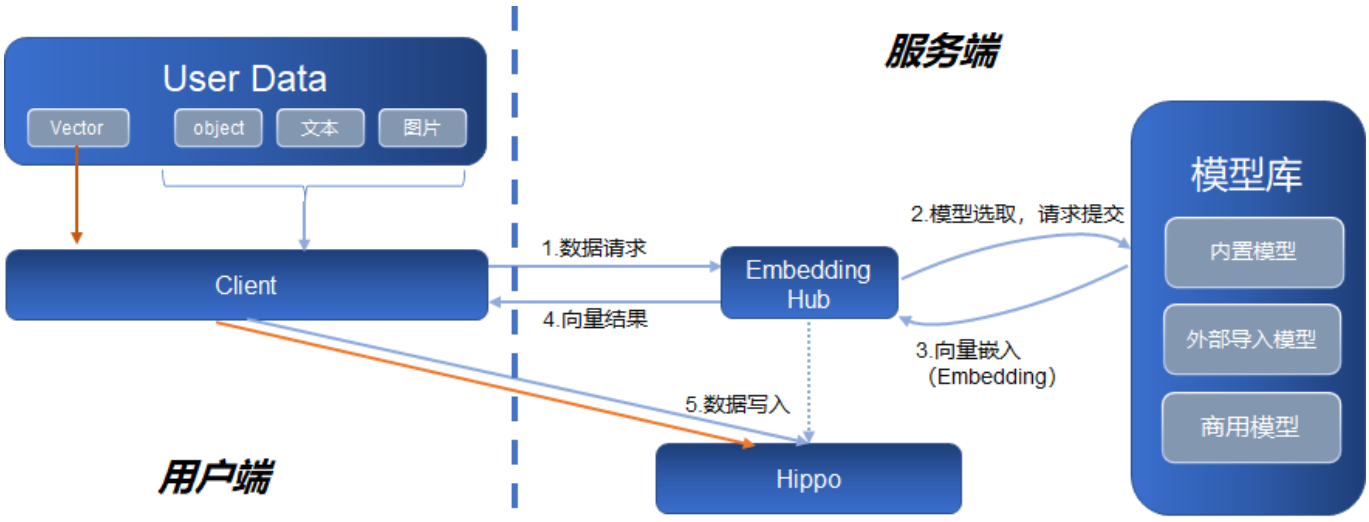
1.2.2. 核心组件

Hippo 的产品功能模块清单及功能介绍如下：

模块名称	模块功能	模块介绍
TDDMS	分布式数据管理系统	TDDMS (Transwarp Distributed Data Management System) 星环自主研发的分布式数据管理系统，管理数据多副本间的强一致或最终一致；管理数据在存储介质上的合理分片分布，扩缩存储容量时，自动管理数据重分布，充分利用存储资源；保障数据高可用，在存储硬件故障时，保持数据存储服务不中断。
Embedding Hub	向量转化工具	Hippo 内置的一个向量转化工具，提供标准化接口连通各类大模型并实现数据的向量嵌入。
Vector Engine	向量搜索引擎	星环自主研发的向量搜索引擎，支持海量向量数据的检索，同时在计算框架上进行了大幅度优化，具备高准确性与高性能的相似检索能力。
Sophon Model Cube	模型仓库	统一了模型生命周期中的模型上架、模型评估和模型部署。可以纳管多模态、多类型的模型，可提高模型的可维护性和可操作性。管理多源异构模型，支持镜像模型、文件模型和组合模型三种类型，并提供模型评估和模型体验功能，最大化模型价值

图表 2 Hippo 核心组件清单

1.2.3. 业务流程介绍



图表 3 Hippo 业务流程图

上图所示，为 Hippo 基本的数据接入流程。在该版本中我们引入了一个新的模块 Embedding Hub，该模块可以有效地处理客户数据的向量转化需求。Hippo 本身支持调用 Hippo Client 将所需的向量数据写入到 Hippo 中；同时，基于 Embedding Hub，我们可以将未处理的各类文本、图片等非结构化数据转发到所配置关联的模型中，并将模型处理后的向量结果返回并写入，降低客户自己处理数据的开销。Embedding Hub 也支持多样化的模型选择，可以选择 Sophon

Model Cube 模型库中的各类模型，也可以使用各类开源或者商用模型，仅需在该模块的配置文件预先配置好链接方式即可。

1.3. 产品优势

优势概括	细节描述
云原生系统	Hippo 采用全面容器化部署，支持服务的弹性扩缩容；同时具备多租户和强大的资源管控能力。
分布式部署	Hippo 具备分布式部署能力，有丰富的大规模集群部署经验；Hippo 通过 Raft 算法确保数据的强一致性；同时提供故障迁移，数据修复等数据保障能力。
数据多模	Hippo 可与 TDH 结构化存储、非结构化存储数据打通，实现数据联合检索。
高性能检索	Hippo 支持多进程架构与 GPU 加速，可以充分发挥并行检索能力；同时支持多类索引，满足不同业务场景；支持检索速度和内存使用的特定优化，支持寄存器级算法优化。
接口多样化	提供标准的 Python、Restful、Java API。

图表 4 Hippo 产品优势

1.4. 管理组件

模块名称	模块功能	模块介绍
Transwarp Aquila	智能运维分析平台软件	Transwarp Aquila 作为一站式综合运维平台，提供监控仪表盘、告警通知、日志生命周期管理、日志检索、审计日志等功能。 同时，Aquila 还预置了基础监控、大数据监控、PaaS 层监控等多维度的监控资源，能做到整个 Hippo 产品各个维度的开箱即用的一键运维。
Transwarp Manager	大数据管理软件	Transwarp Manager 是负责配置、管理和运维 Hippo 集群的图形化工具。用户只需通过几个手动步骤，就可以在 X86、ARM、MIPS 等各架构服务器或基于 Docker 的云端平台上完成集群部署，并且提供告警、健康检测、监控和度量等运维服务。用户可以实时的浏览各服务的状态，并且在告警出现时采取恰当的措施以处理应对。此外，Manager 还提供了一些便捷的运维功能，例如磁盘管理、软件升级和服务迁移等。
Transwarp Cloud Operating System	云原生操作系统	TCOS 是星环自主研发的云原生操作系统，提供统一的资源调度框架，通过容器化编排，能够统一调度计算、存储、网络等各基础资源，TCOS 支持主流国产操作系统、国产服务器，并且支持混合部署，满足国产化部署需求。

图表 5 管理组件

2. 安装教程

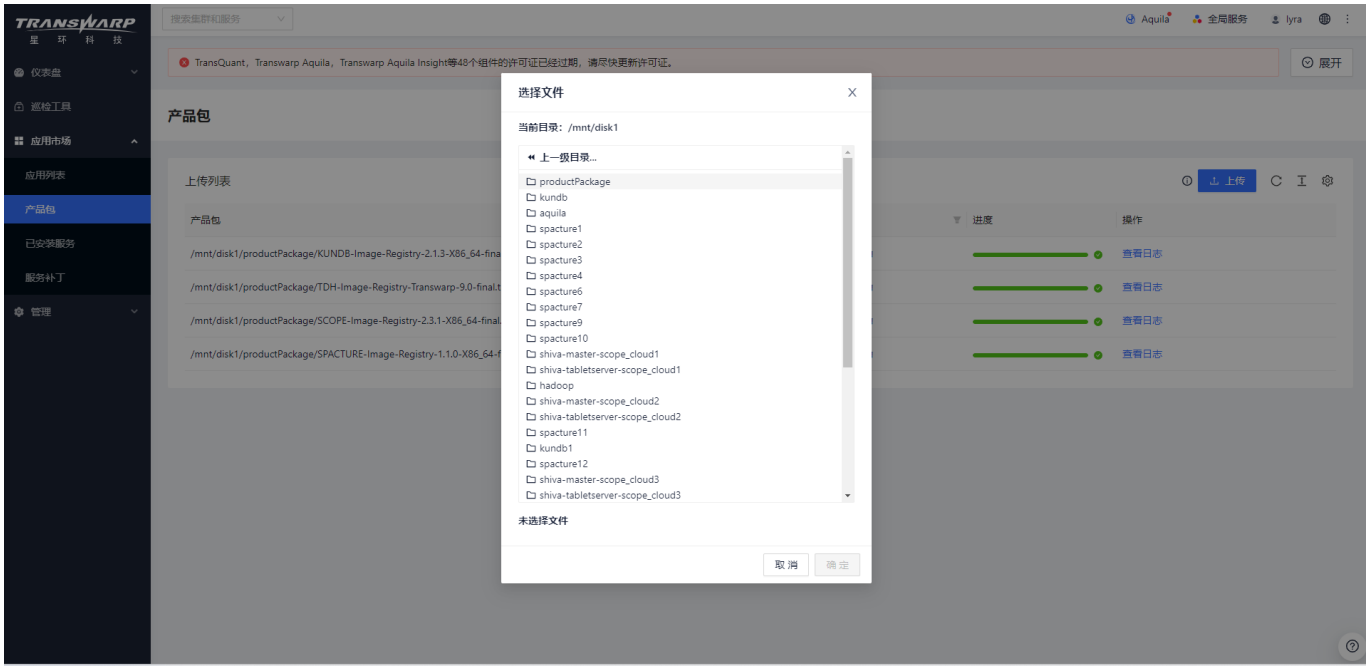
2.1. 安装 Transwarp Data Hub

Hippo 的安装通过 Transwarp Manager 管理平台 (以下简称 Manager) 完成，可以在第一次安装 Transwarp Data Hub 集群时安装，也可以向安装好的集群另外安装 Hippo 服务。Transwarp Data Hub 集群的安装在《Transwarp Data Hub 安装手册》中有详细的介绍，这里不再赘述。

2.2. 安装 Hippo

2.2.1. 上传产品包

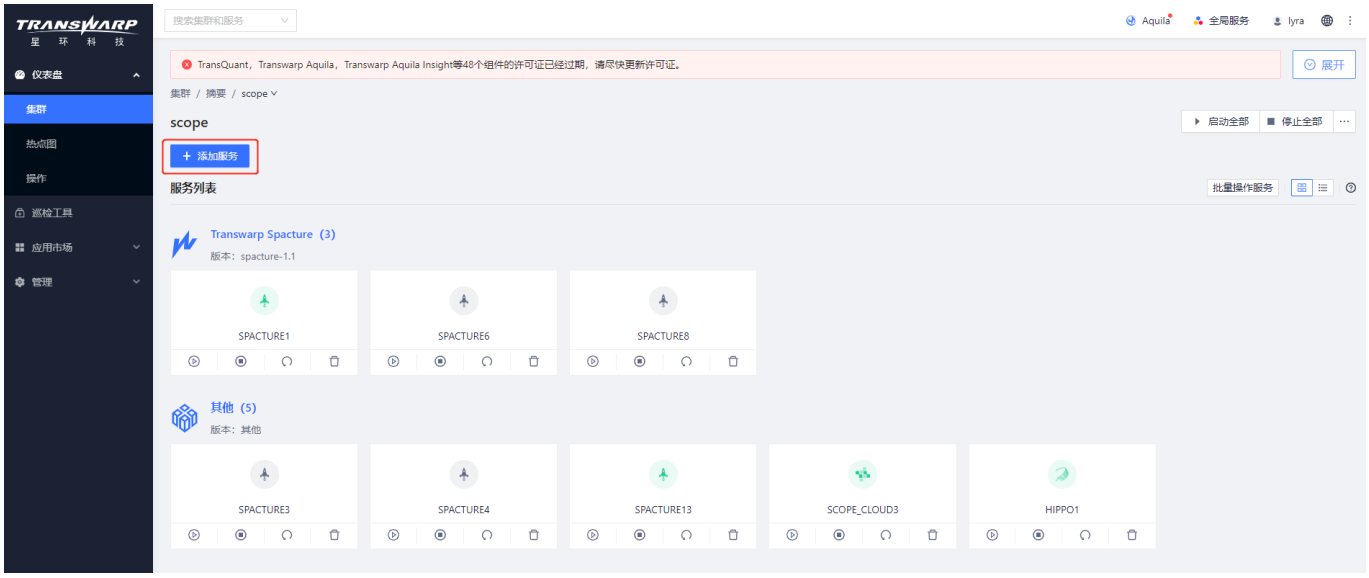
登陆 Manager，点开左侧“应用市场”下拉框，选择“产品包”。在右侧对应页面的右侧点击“上传”，找到 Hippo 产品包所在的目录选择对应产品包，点击“确定”，然后等到上传完成。



图表 6 上传产品包

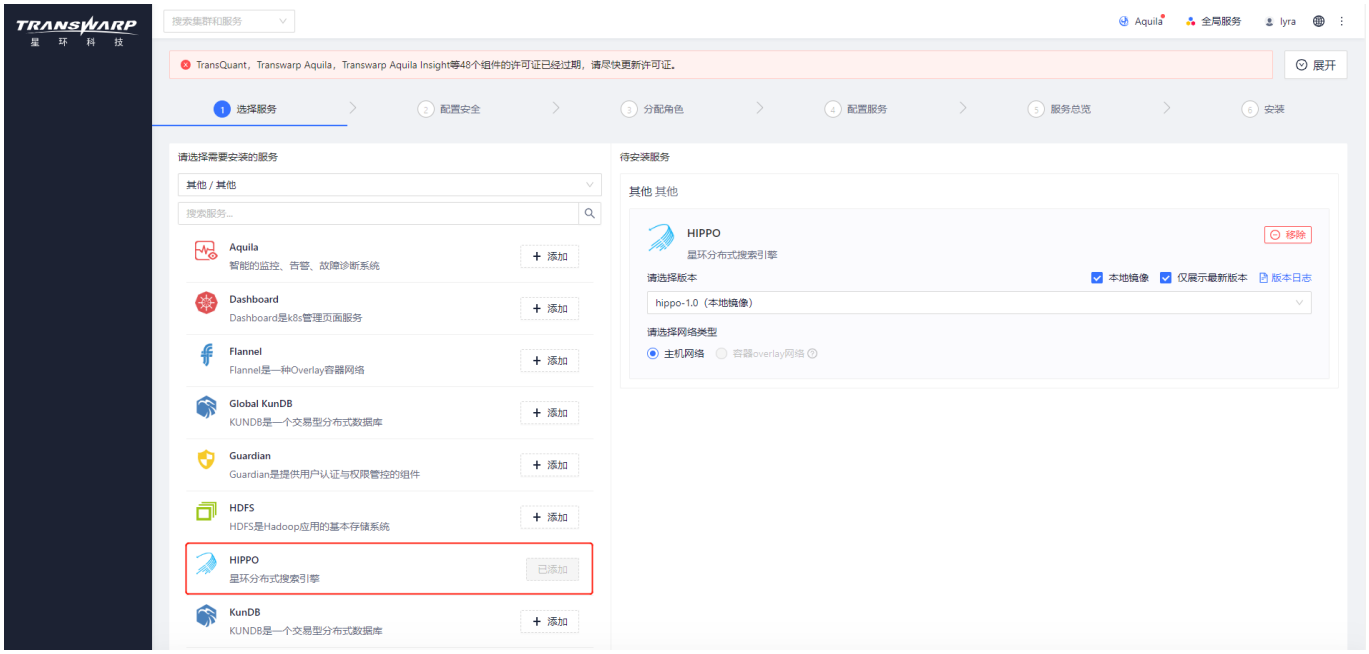
2.2.2. 添加服务

1. 在 Manager 的“集群”页面上，点击“+添加服务”。



图表 7 添加服务

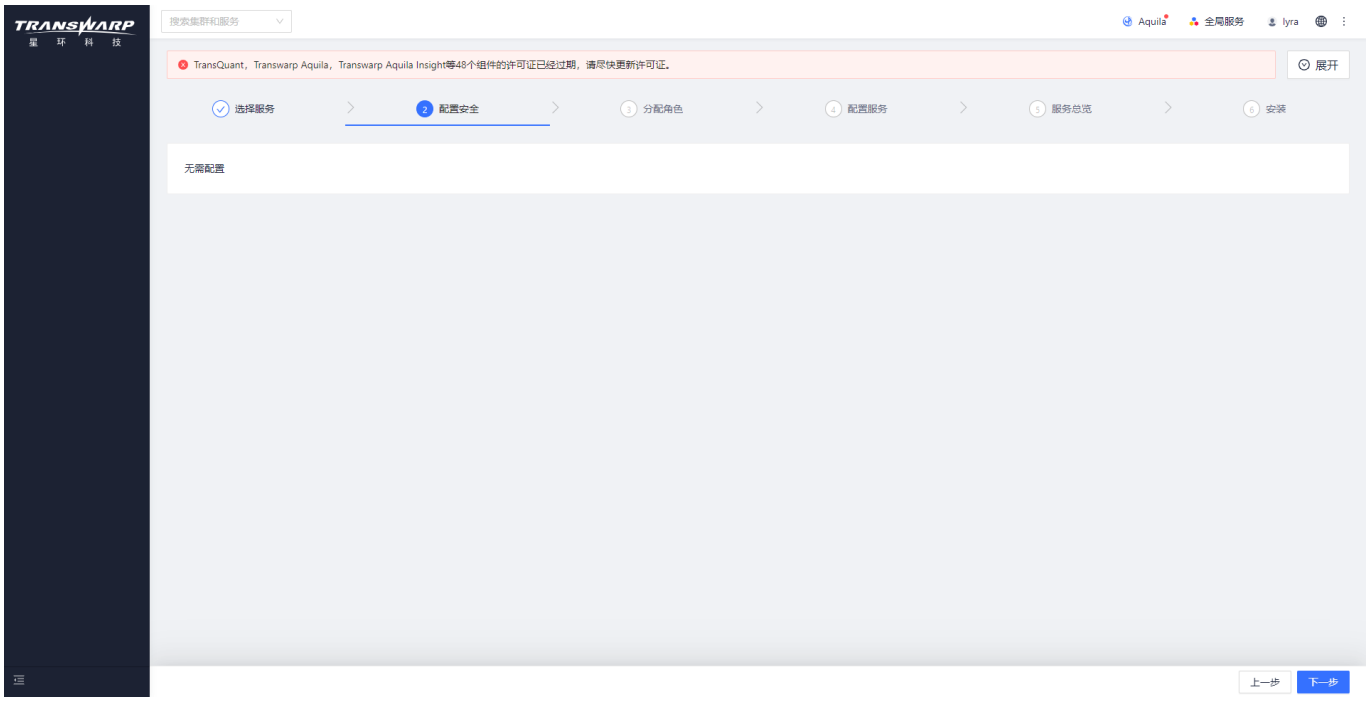
2. 在新弹出的“选择服务”页面，选择“其他/其他”，然后在显示的服务中选择 HIPPO，点击“+添加”。确认服务相关信息无误后，右下角点击“下一步”。



图表 8 选择 Hippo

2.2.3. 配置安全

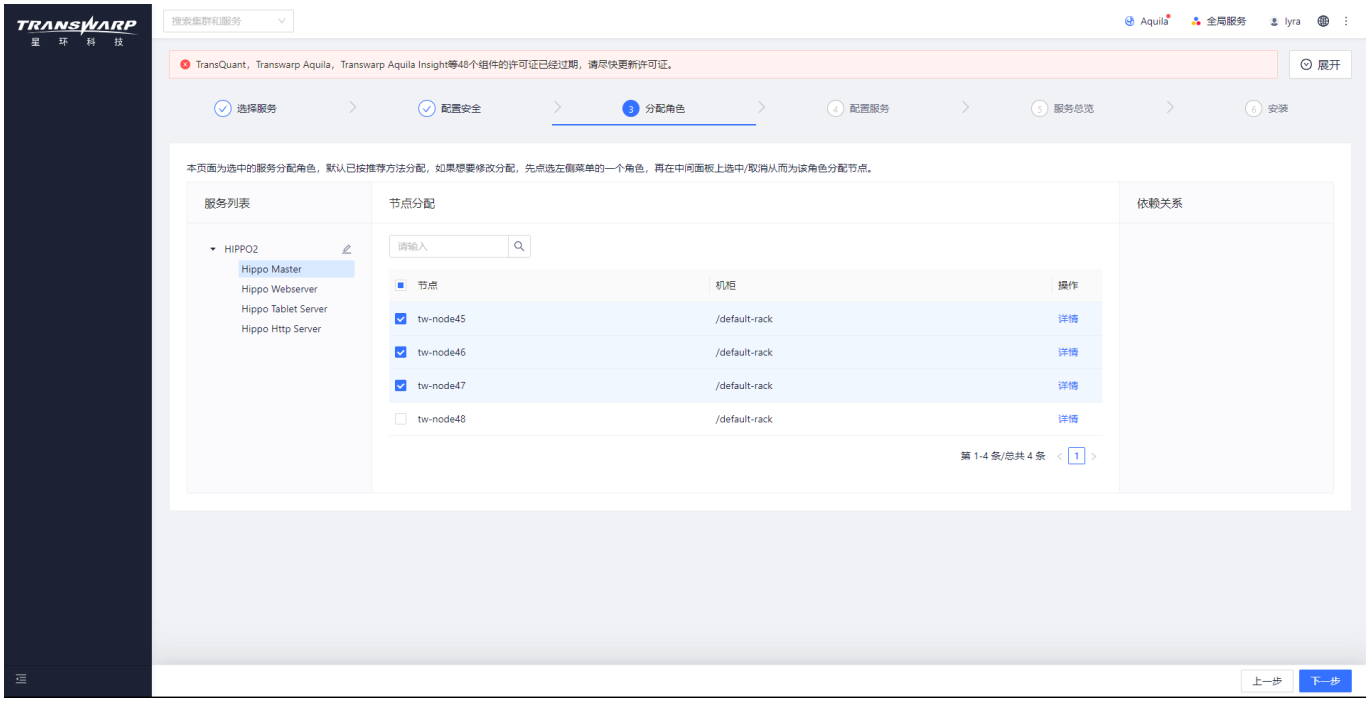
目前 Hippo 服务无需通过 Manager 配置安全，在“配置安全”这一步直接点击“下一步”即可。



图表 9 配置安全

2.2.4. 分配角色

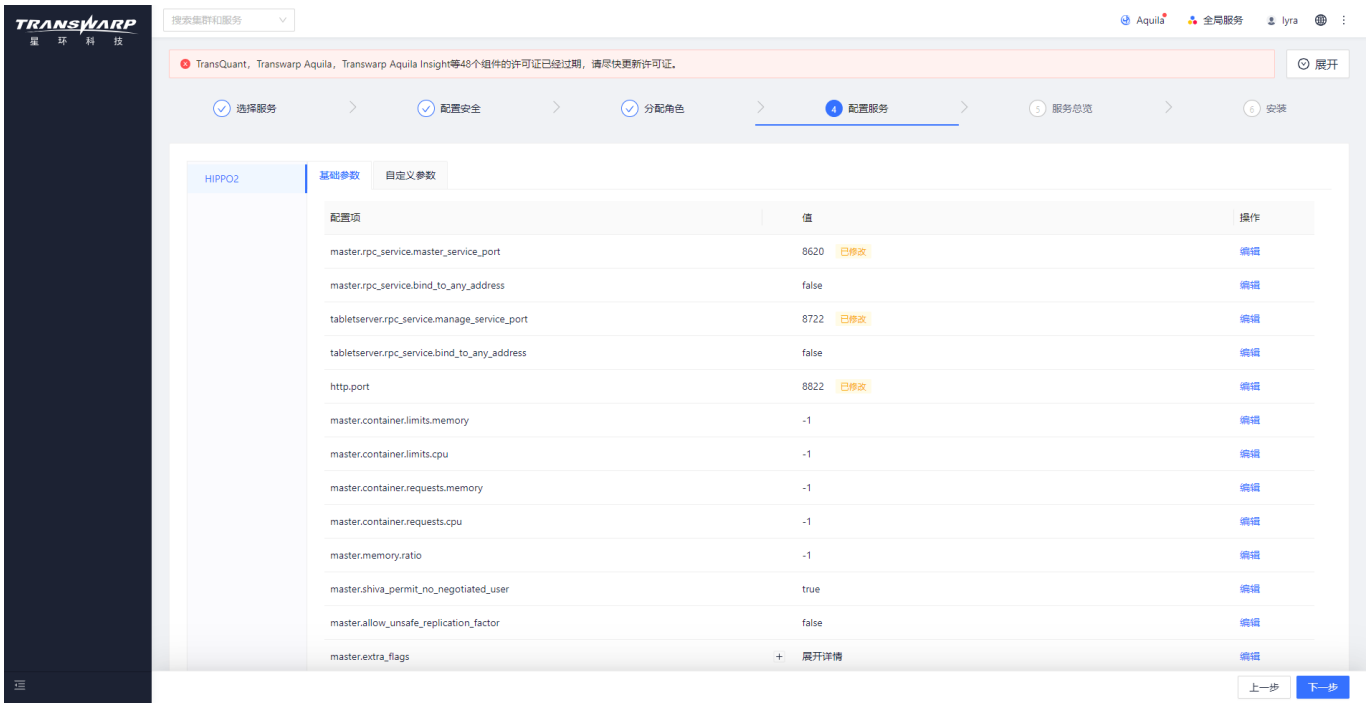
在“分配角色”页面，可以查看即将被安装的 Hippo 服务对应的信息，包括：服务名称，各个角色对应的节点分配情况等。服务名称可自定义，不过需注意不要和已安装的 Hippo 服务名称重复，否则会报错。另外如果角色分配节点与之前已安装的 Hippo 服务也有重复，需注意端口冲突问题。



图表 10 分配角色

2.2.5. 配置服务

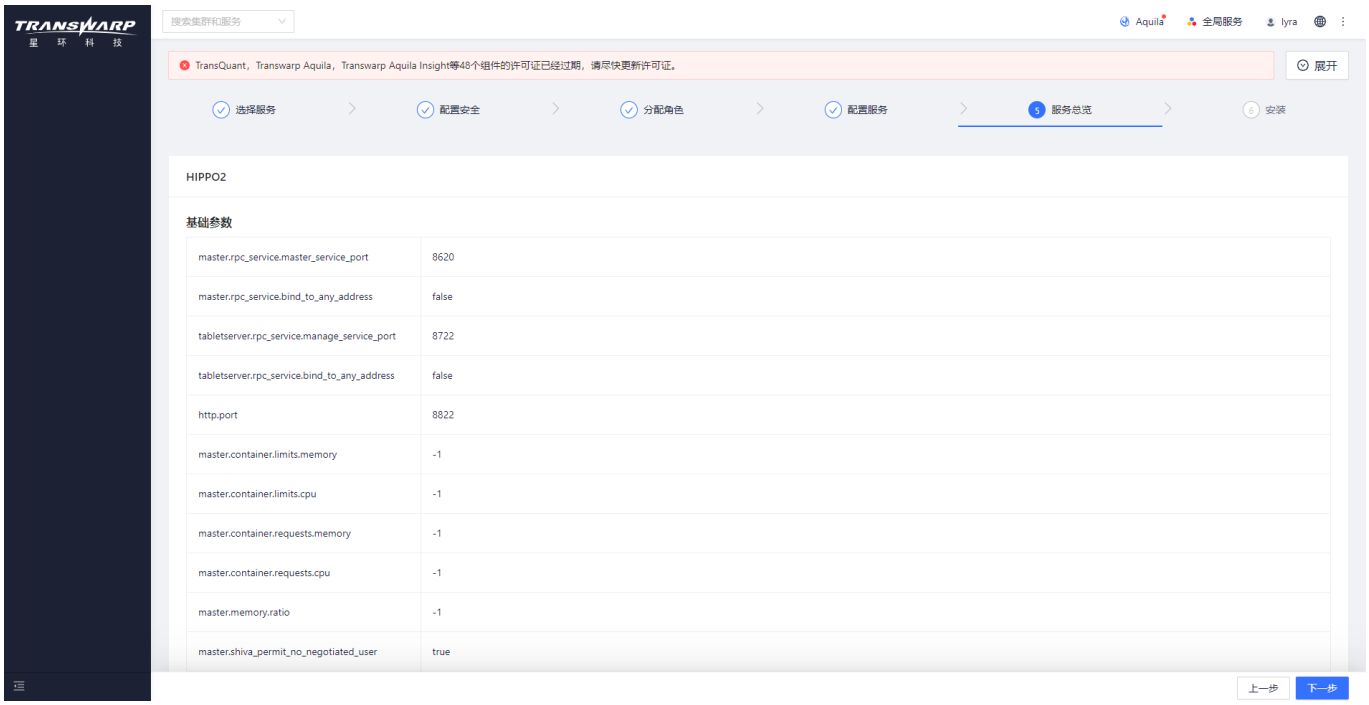
“配置服务”页面显示当前要安装的 Hippo 服务对应的各个配置信息。需注意如果之前已安装了其他 Hippo 服务，并且当前在安装的服务相比之前有角色使用重复的节点，需注意端口冲突问题。用户可在当前页面进行端口修改。有关 Hippo 服务默认端口信息，请参考下方 3.1. 默认端口信息章节。



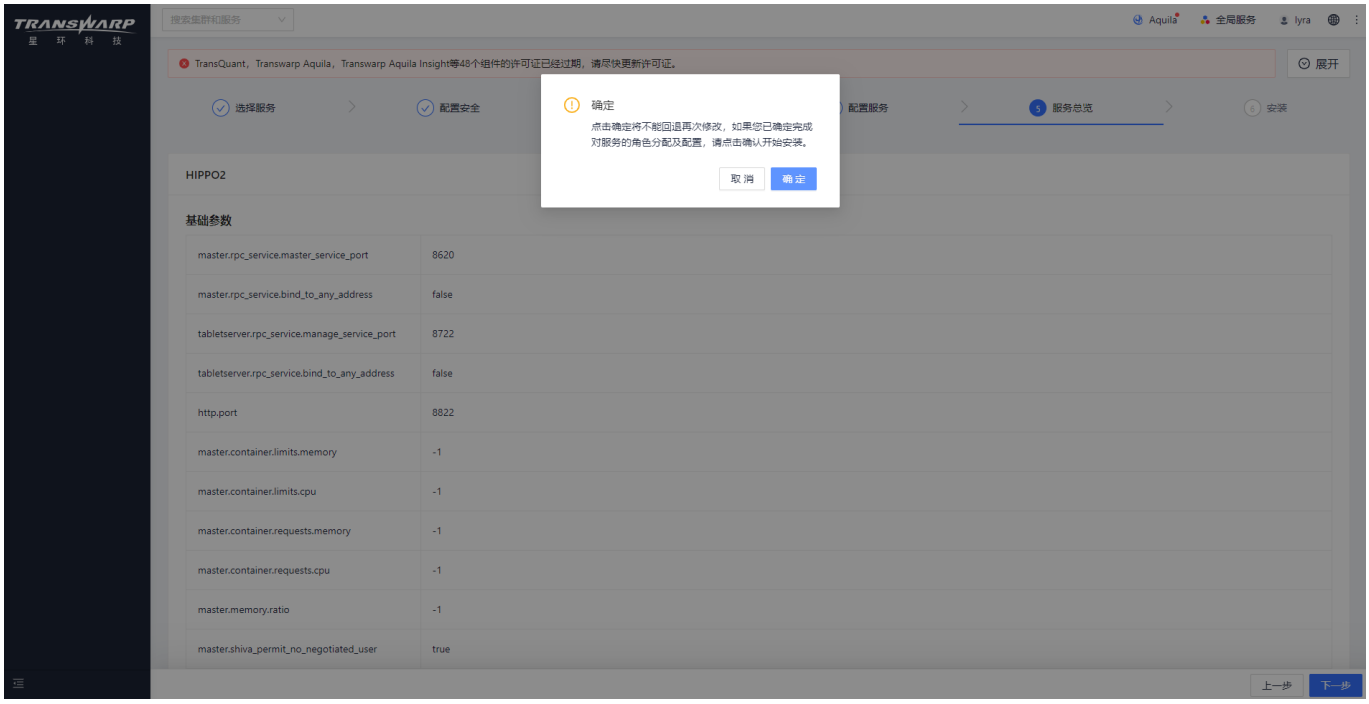
图表 11 配置服务

2.2.6. 服务总览

可在“服务总览”页面检查所有配置信息，如果需要修改，可以点击“上一步”。无需可直接点击“下一步”，点击“确定”进入“安装”页面。



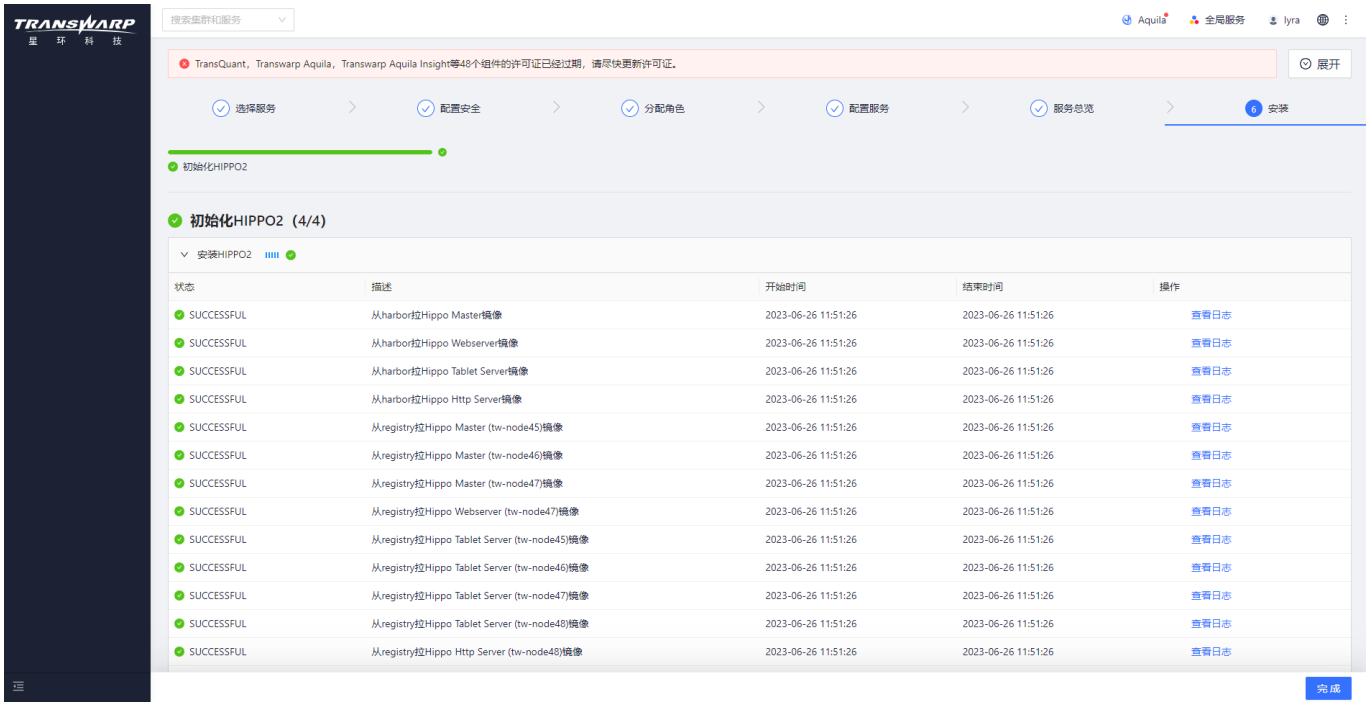
图表 12 服务总览



图表 13 确定安装

2.2.7. 安装服务

在“安装”界面等待服务安装完成，如遇到问题，可点击报错一行的“查看日志”选项确认更多报错信息。



图表 14 安装服务

3. 连接数据库

3.1. 默认端口信息

Hippo 默认端口信息如下：

配置项	端口说明	配置地址	端口号
http.port	Webserver 端口	Manager 界面 / 后台配置路径：/etc/hippo/conf/shiva-restful.sh	8902
httpserver.port	Restful API 端口	Manager 界面 / 后台配置路径：/etc/hippo/conf/elasticsearch.yml	8902
master.rpc_service.master_service_port	Shiva Master 端口	Manager 界面 / 后台配置路径：/etc/hippo/conf/master.conf	8630 - 8633
tabletserver.rpc_service.manage_service_port	Tablet Server 端口	Manager 界面 / 后台配置路径：/etc/hippo/conf/tabletserver.conf	8702 - 8705

图表 15 Hippo 默认端口信息

3.2. 连接方式

Hippo 支持通过 Restful API，Python 以及 Java 等多种方式连接，以下是 Restful API 的参考 Demo。

3.2.1. Restful API

```
curl -u shiva:shiva -XPUT 'ip:port/hippo/v1/command?pretty' -H 'Content-
Type: application/json' -d'{}'
```

参数说明：

参数	描述	示例
-u	指定操作 Hippo 的用户和密码	shiva:shiva
-XPUT	请求类型	
ip	Hippo HTTP Server 所在节点	
port	Hippo HTTP Server 对应端口	默认 8902
url	可参考第 5 章节 Hippo 相关操作介绍，用于具体执行的每个命令	

图表 16 Restful API 连接参数说明

4. 角色介绍

1. Hippo Master
   - 又称为 Shiva Master。
   - 负责 TDDMS 的元信息存储
   - Master 是一个 Raft Group，由多个 Master 节点组成
   - 部署中推荐配置三或五个 Master 节点
2. Hippo Tablet Server
   - 又称为 Shiva Tablet Server
   - 负责 TDDMS 的数据存储
   - 部署中推荐至少存在五个 Tablet Server 节点
3. Hippo Webserver
   - Hippo 内置的一个轻量级监控组件，用于监控集群/索引状态
   - 提供 Rest 风格的界面化操作平台
   - 部署中至少需要一个 Webserver 节点
4. Hippo HTTP Server
   - 负责 Hippo 的 HTTP Restful 请求的交互
   - 支持 HTTP/Java/Python
   - 部署中至少部署一个，可部署多个

5. 使用指南

与传统关系型数据库类似，使用者可以在 Hippo 创建 Database，Database 的权限管理与表类似，用户对 Database 进行不同操作需要具有对应的权限。

5.1. 数据库相关操作

5.1.1. 创建数据库

Hippo 集群自带一个名为 default 的数据库，默认情况下所有操作在 default 数据库下进行，使用以下命令创建新的数据库，库名为 book。

Restful API：

```
curl -u shiva:shiva -XPUT 'localhost:8902/hippo/v1/_database?pretty' -H 'Content-
Type: application/json' -d'{
  "database_name" : "book"
}';
```

返回结果：

```
{
  "acknowledged" : true
}
```

5.1.2. 列出数据库

使用以下命令列出集群中当前所有数据库信息。

Restful API：

```
curl -u shiva:shiva -XPUT 'localhost:8902/hippo/v1/_database?pretty' -H 'Content-
Type: application/json' -d'{
  "database_name" : "book"
}';
```

返回结果：

```
{
  "acknowledged" : true
}
```

5.1.3. 删除数据库

删除数据库时，需要首先删除数据库中所有表，包括处于回收站中的表，否则删除数据库操作将提示错误。

Restful API：

```
curl -u shiva:shiva -XDELETE 'localhost:8902/hippo/v1/_database?pretty' -H 'Content-
Type: application/json' -d'{
  "database_name" : "book"
}';
```

返回结果：

```
{
  "acknowledged" : true
}
```

5.2. 表相关操作

5.2.1. 建表

本节介绍如何在 Hippo 中建表。在以下示例中，我们将创建一个 1 分片 1 副本、名为 book 的表，book 表包含一个 int64 类型的主键列 book_id，一个 int64 类型的标量列 word_count，以及一个维度为 2 的浮点向量列 book_intro。

Restful API：

```
curl -u shiva:shiva -XPUT 'localhost:8902/hippo/v1/{table}?pretty' -H 'Content-
Type: application/json' -d'{
  "settings": {
    "number_of_shards" : 1,
    "number_of_replicas" : 1
  },
  "schema": {
    "fields": [
      {
        "name": "book_id",
        "is_primary_key": true,
        "data_type": "int64"
      },
      {
        "name": "word_count",
        "is_primary_key": false,
        "data_type": "int64"
      },
      {
        "name": "book_intro",
        "data_type": "float_vector",
        "is_primary_key": false,
        "type_params": {
          "dimension" : 2
        }
      }
    ]
  }
}';
```

返回结果：

```
{
  "acknowledged" : true
}
```

参数说明：

参数	描述	选项
table	表名	
database	数据库名，URL 参数	[1， 5]
fields	新建 schema 的列集	
name (field)	字段名	N/A
is_primary_key	判断是否为主键	true / false
data_type	字段对应数据类型	• int8 • int16 • int32 • int64 • float • double • bool • string • char • varchar • varchar2 • date • timestamp • datetime • binary • blob • clob • float_vector
dim (type_params)	向量维度	[1, 65， 536]

图表 17 建表操作参数说明

5.2.2. 表重命名

Hippo 支持对表进行重命名。

Restful API：

```
curl -u shiva:shiva -XPUT 'localhost:8902/hippo/v1/_rename_table?pretty' -H 'Content-
Type: application/json' -d'{
  "database_name" : "book",
  "table_name" : "book",
```

```
  "new_table_name" : "new_book"
}';
```

返回结果：

```
{
  "acknowledged" : true
}
```

参数说明：

参数	描述	是否必选
table_name	需要进行重命名的表名	是
new_table_name	新的表名	是
database_name	需要进行重命名表所在数据库	否，默认为 default

图表 18 表重命名操作参数说明

5.2.3. 删表

使用以下命令从 Hippo 中删除表，被删除的表会暂时处于回收站中，并在一定时间后自动清理。

Restful API

```
curl -u shiva:shiva -XDELETE 'localhost:8902/hippo/v1/book' // 支持通配
```

返回结果：

```
{
  "acknowledged" : true
}
```

参数说明：

参数	描述	是否必选
table	待删除表名，支持以*通配，支持指定多个 Pattern，多个 Pattern 以逗号分割	是
database_name	待删除表所在数据库	否，默认为 default

图表 19 删表操作参数说明

5.2.4. 查看表信息

本节描述如何获取 Hippo 表信息。

5.2.4.1. 检查表是否存在

Restful API：

```
curl -u shiva:shiva -XGET 'localhost:8902/hippo/v1/_check_table_existence?pretty' -
H 'Content-Type: application/json' -d'{
  "database_name" : "book",
  "table_name" : "book"
}';
```

返回结果：

```
{
  "acknowledged" : true
}
```

参数说明：

参数	描述	是否必选
table_name	表名	是
database_name	表所在 database	否，默认为 default

图表 20 检查表存在操作参数说明

5.2.4.2. 获取表详细信息

Restful API：

```
curl -u shiva:shiva -XGET 'localhost:8902/hippo/v1/{table}?pretty'
```

返回结果：

```
{
  "book" : {
    "id" : "b41cab6e3418434a9d820f0405263303",
    "database" : "default",
    "settings" : {
      "number_of_shards" : 1,
      "creation_date" : "2023-05-21 22:22:13",
      "number_of_replicas" : 1,
      "data_center" : "DEFAULT",
      "dc_affinity" : false,
      "disaster_preparedness" : false,
      "scatter_replica" : false
    },
    "schema" : {
      "fields" : [
        {
          "name" : "book_id",
          "id" : 0,
          "is_primary_key" : true,
          "is_nullable" : false,
```

```
        "data_type" : "BIGINT"
      },
      {
        "name" : "word_count",
        "id" : 1,
        "is_primary_key" : false,
        "is_nullable" : true,
        "data_type" : "BIGINT"
      },
      {
        "name" : "book_intro",
        "id" : 2,
        "is_primary_key" : false,
        "is_nullable" : true,
        "data_type" : "FLOAT_VECTOR",
        "type_params" : {
          "dimension" : 2
        }
      },
      {
        "name" : "__schemaversion__",
        "id" : 3,
        "is_primary_key" : false,
        "is_nullable" : true,
        "data_type" : "BIGINT"
      },
      {
        "name" : "__rowid__",
        "id" : 4,
        "is_primary_key" : false,
        "is_nullable" : true,
        "data_type" : "BIGINT"
      }
    ]
  },
  "embedding_indexes" : [
    {
      "name" : "book_intro_index",
      "id" : 0,
      "column" : "book_intro",
      "index_type" : "IVF_FLAT",
      "metric_type" : "L2",
      "params" : {
        "nlist" : 10
```

```
        }
      }
    ]
  }
}
```

参数说明：

参数	描述	是否必选
table	表示待匹配的表名，支持以*通配，支持指定多个 Pattern，多个 Pattern 以逗号分割	是
database_name	表所在 database	否，默认为 default

图表 21 获取表详细信息操作参数说明

5.2.4.3. 列出数据库中的表

使用以下命令列出数据库中的表。该命令会返回表名、表状态、id 以及一些基本的表级统计信息。

Restful API：

```
curl -u shiva:shiva -XGET 'localhost:8902/hippo/v1/_cat/tables/{table}?v'
curl -u shiva:shiva -XGET 'localhost:8902/hippo/v1/_cat/tables?v'
```

返回结果：

```
[root@tw-node45 ~]# curl -u shiva:shiva tw-node45:8902/hippo/v1/_cat/tables?v
epoch      timestamp status table                                                  uuid                             pri rep sst mem.reader mem.memtable embedding.segments embedding.in_memory
1687774872 10:21:12  open   default#hippo-deep-image-dim_25-metr_angular           f6c678d9805c46cc9d5df97b273e9f0c   1   3 9   544        8192         2                1183400
1687774872 10:21:12  open   default#hippo-glove-dim_100-metr_angular              319137871dec4bb3a86a808affdf530a   1   3 31  1600       8192         0                0
1687774872 10:21:12  open   default#book                                         63ccee1440594e47abb070a307a14933   1   1 8   608        8192         1                101
1687774872 10:21:12  open   default#hippo-deep-dim_96-metr_angular              e7678dee80b24a37a3f691fb6f1124f1   1   3 256 12512      8192         0                0
1687774872 10:21:12  open   default#hippo-glove-dim_200-metr_angular            f031e3f8b4164a4ca2b88af8b6f389ce   1   3 62  3424       8192         2                1183400
1687774872 10:21:12  open   default#hippo-fashion-mnist-dim_784-metr_euclidean  58cdb30771934c4bbb1ee51fa775e4f5   1   3 11  640        8192         0                0
1687774872 10:21:12  open   default#book1                                        6ff476b55ca14b4784fdbfe89225d6e3   1   1 3   256        8192         1                100
1687774872 10:21:12  open   default#hippo-gist-dim_960-metr_euclidean           a59ea7db4ec34dfcbd247a1e685def49   1   3 231 11200      8192         2                1000000
1687774872 10:21:12  open   default#hippo-deep-image-dim_100-metr_angular_2023-06-21-14-33-10 12b14a7c46a44abf901c02c03c624fca   3   3 9   768        24576        3                4200
1687774872 10:21:12  open   default#hippo-deep-image-dim_100-metr_angular_2023-06-21-14-38-42 0ab28daf73b44f9cbc60489508f3c961   3   3 45  2496       24576        6                2072000
1687774872 10:21:12  open   default#hippo-sift-dim_128-metr_euclidean           31ae47aaa0fd41b6953863ec00a02c51   1   3 37  2112       8192         0                0
```

图表 22 通过 Restful API 列举表操作示例图

参数说明：

参数	描述	是否必选
table	表示待匹配的表名，支持以*通配，支持指定多个 Pattern，多个 Pattern 以逗号分割	否，默认为*，表示列出数据库中的所有表
database_name	表所在 database	否，默认为 default

图表 23 通过 Restful API 列举表操作参数说明

5.2.4.4. 列出数据库中的分片

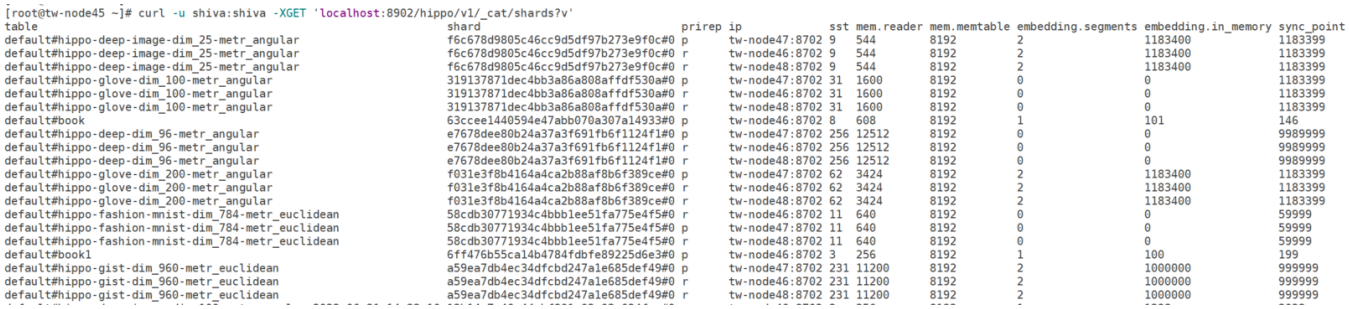
该命令会返回分片信息，以及一些基本的分片级统计信息。

Restful API：

```
curl -u shiva:shiva -XGET 'localhost:8902/hippo/v1/_cat/shards/{table}?v'
```

```
curl -u shiva:shiva -XGET 'localhost:8902/hippo/v1/_cat/shards?v' // 默认列出全部Hippo 表
```

返回结果：



```
[root@tw-node45 ~]# curl -u shiva:shiva -XGET 'localhost:8902/hippo/v1/_cat/shards?v'
table                                                shard                                pri rep ip              sst mem.reader mem.memtable embedding.segments embedding.in_memory sync_point
default#hippo-deep-image-dim_25-metr_angular         f6c678d9805c46cc9d5df97b273e9f0c#0 p   tw-node47:8702 9   544        8192         2                  1183400             1183399
default#hippo-deep-image-dim_25-metr_angular         f6c678d9805c46cc9d5df97b273e9f0c#0 r   tw-node46:8702 9   544        8192         2                  1183400             1183399
default#hippo-deep-image-dim_25-metr_angular         f6c678d9805c46cc9d5df97b273e9f0c#0 r   tw-node48:8702 9   544        8192         2                  1183400             1183399
default#hippo-glove-dim_100-metr_angular             319137871dec4bb3a86a808affdf530a#0 p   tw-node47:8702 31  1600       8192         0                  0                   1183399
default#hippo-glove-dim_100-metr_angular             319137871dec4bb3a86a808affdf530a#0 r   tw-node46:8702 31  1600       8192         0                  0                   1183399
default#hippo-glove-dim_100-metr_angular             319137871dec4bb3a86a808affdf530a#0 r   tw-node48:8702 31  1600       8192         0                  0                   1183399
default#book                                         63ccee1440594e47abb070a307a14933#0 p   tw-node46:8702 8   608        8192         1                  101                 146
default#hippo-deep-dim_96-metr_angular               e7678dee80b24a37a3f691fb6f1124f1#0 p   tw-node47:8702 256 12512      8192         0                  0                   9989999
default#hippo-deep-dim_96-metr_angular               e7678dee80b24a37a3f691fb6f1124f1#0 r   tw-node46:8702 256 12512      8192         0                  0                   9989999
default#hippo-deep-dim_96-metr_angular               e7678dee80b24a37a3f691fb6f1124f1#0 r   tw-node48:8702 256 12512      8192         0                  0                   9989999
default#hippo-glove-dim_200-metr_angular             f031e3f8b4164a4ca2b88af8b6f389ce#0 p   tw-node47:8702 62  3424       8192         2                  1183400             1183399
default#hippo-glove-dim_200-metr_angular             f031e3f8b4164a4ca2b88af8b6f389ce#0 r   tw-node46:8702 62  3424       8192         2                  1183400             1183399
default#hippo-glove-dim_200-metr_angular             f031e3f8b4164a4ca2b88af8b6f389ce#0 r   tw-node48:8702 62  3424       8192         2                  1183400             1183399
default#hippo-fashion-mnist-dim_784-metr_euclidean   58cdb30771934c4bbb1ee51fa775e4f5#0 r   tw-node46:8702 11  640        8192         0                  0                   59999
default#hippo-fashion-mnist-dim_784-metr_euclidean   58cdb30771934c4bbb1ee51fa775e4f5#0 p   tw-node47:8702 11  640        8192         0                  0                   59999
default#hippo-fashion-mnist-dim_784-metr_euclidean   58cdb30771934c4bbb1ee51fa775e4f5#0 r   tw-node48:8702 11  640        8192         0                  0                   59999
default#book1                                        6ff476b55ca14b4784fdbfe89225d6e3#0 p   tw-node46:8702 3   256        8192         1                  100                 199
default#hippo-gist-dim_960-metr_euclidean            a59ea7db4ec34dfcbd247a1e685def49#0 p   tw-node47:8702 231 11200      8192         2                  1000000             999999
default#hippo-gist-dim_960-metr_euclidean            a59ea7db4ec34dfcbd247a1e685def49#0 r   tw-node46:8702 231 11200      8192         2                  1000000             999999
default#hippo-gist-dim_960-metr_euclidean            a59ea7db4ec34dfcbd247a1e685def49#0 r   tw-node48:8702 231 11200      8192         2                  1000000             999999
```

图表 24 通过 Restful API 列举数据库分片操作示例图

参数说明：

参数	描述	是否必选
table	表示待匹配的表名，支持以*通配，支持指定多个 Pattern，多个 Pattern 以逗号分割	否，默认为*，表示列出数据库中的所有表
database_name	database 名	否，默认为 default

图表 25 通过 Restful API 列举数据库分片操作参数说明

5.2.5. 表配置

Hippo 表配置信息包括:

配置项	描述	可选项	是否可更新
number_of_shards	分片数	[1,5]	否
number_of_replicas	副本数	大于 0	是
creation_date	创建时间		否
data_center	跨数据中心相关配置，表示主数据中心		是
disaster_preparedness	跨数据中心相关配置，表示是否需要跨数据中心高可用	True 或 False	是
scatter_replica	跨数据中心相关配置，设置为 true 时，对 5 副本 3 中心，hippo 会确保副本以 221 形式分布在 3 个数据中心	True 或 False	是
dc_affinity	跨数据中心相关配置，如设置为 true，hippo 会尽量将副本的 leadership 迁移到主中心	True 或 False	是
tag	标签		是
embedding.segment.max_size_mb	向量索引每个段预期大小	默认 512MB	是
embedding.segment.seal_proportion	当内存中的 flat 索引大小超过 embedding.segment.max_size_mb 的一定比例时，hippo 会自动为增量数据生成向量索引，本配置控制这个比例	默认 0.2	是

embedding.segment.max_deletion_proportion	当一个向量索引段中被删除数据数量超过一定比例时，hippo 会自动合并改向量索引段，以释放资源，本配置控制这个比例	默认 0.1	是

图表 26 Hippo 表配置参数说明

5.2.5.1. 查看表配置

使用以下命令查看表配置。

Restful API：

```
curl -u shiva:shiva -XGET 'localhost:8902/hippo/v1/_settings?pretty'
curl -u shiva:shiva -XGET 'localhost:8902/hippo/v1/{table}/_settings?pretty'
curl -u shiva:shiva -XGET 'localhost:8902/hippo/v1/{table}/_setting?pretty'
```

返回结果：

```
{
  "default#book" : {
    "number_of_shards" : 1,
    "creation_date" : "2023-06-15 11:42:25",
    "number_of_replicas" : 1,
    "data_center" : "DEFAULT",
    "dc_affinity" : false,
    "disaster_preparedness" : false,
    "scatter_replica" : false
  }
}
```

参数说明：

参数	描述	是否必选
table	表示待匹配的表名，支持以*通配，支持指定多个 Pattern，多个 Pattern 以逗号分割	否，默认为*，获取 database 下所有表信息
database_name	表所处 database	否，默认为 default

图表 27 查看表配置操作参数说明

5.2.5.2. 更新表配置

使用以下命令更新表配置。

Restful API：

```
curl -u shiva:shiva -XPUT 'localhost:8902/hippo/v1/{table}/_settings?pretty' -H 'Content-Type: application/json' -d'{
  "number_of_replicas" : 1,
```

```
  "data_center" : "beijing",
  "disaster_preparedness" : true,
  "scatter_replica" : true,
  "dc_affinity" : true,
  "tag" : "newTag",
  "tag.clear" : true,
  "embedding.segment.max_size_mb" : 512,
  "embedding.segment.seal_proportion" : 0.2,
  "embedding.segment.max_deletion_proportion" : 0.1
}';
```

返回结果：

```
{
  "acknowledged" : true
}
```

参数说明：

参数	描述	是否必选
table	表示待匹配的表名，支持以*通配，支持指定多个 Pattern，多个 Pattern 以逗号分割	是
database_name	表所处 database	否，默认为 default

图表 28 更新表配置操作参数说明

5.2.6. 别名

Hippo 支持表别名，用户可以像访问表一样访问别名。别名的一个典型应用场景是，当需要更改一个表的 Embedding 算法时，用户需要对所有数据的向量列进行更新，这个过程往往比较耗时，为了避免对前端查询造成影响，用户可以使用别名。具体做法是，前端业务使用别名进行查询，更新 Embedding 算法时，生成一个新的表写入新的向量数据，完成后将别名指向新表。本节介绍 Hippo 的别名相关操作。

5.2.6.1. 别名操作

当别名创建后，对表进行重命名并不会影响别名。

Restful API：

```
curl -u shiva:shiva -XPOST 'localhost:8902/hippo/v1/_aliases?pretty' -H 'Content-
Type: application/json' -d'{
  "actions" : [
    {
      "type" : "Add",
      "alias_name" : "book_alias",
      "table_name" : "book"
```

```
    }
  ]
}'
```

返回结果：

```
{
  "acknowledged" : true
}
```

参数说明：

参数	描述	可选项
actions	定义针对别名的一组操作	
type	别名操作类型，表示将表添加到别名，或从别名中删除	Add 或 Remove
alias_name	待操作的别名名字	
table_name	待操作表名	
is_write_table	控制当通过别名写入时，写入的表，如别名中只有一张表，则别名的读写均转向这张表，否则，写请求将转向定义 is_write_table 为 true 的表，读请求就转向别名中的第一张表。别名中只允许有一张表定义 is_write_table 为 true	True 或 false
database_name	表所属 database	

图表 29 别名操作参数说明

5.2.6.2. 删除别名

当别名中的所有表都被从别名中删除后，别名将会被自动删除，也可使用以下命令直接删除一个别名。

Restful API：

```
curl -u shiva:shiva -XDELETE 'localhost:8902/hippo/v1/_aliases/{alias}?pretty'
```

返回结果：

```
{
  "acknowledged" : true
}
```

参数说明：

参数	描述	是否必选
alias	待删除别名	是

图表 30 删除别名操作参数说明

5.3. 写入类操作

本节介绍 Hippo 表写入相关操作。Hippo 会返回写入成功数据的下标以及总共写入成功的数据条数，如果出现行级错误 (比如主键冲突)，Hippo 会返回具体的行级错误。

5.3.1. 插入

本节介绍如何向 Hippo 中插入数据。

Restful API：

```
curl -u shiva:shiva -XPUT 'localhost:8902/hippo/v1/{table}/_bulk?pretty' -H 'Content-
Type: application/json' -d'{
  "fields_data": [
    {
      "field_name": "book_id",
      "field": [
1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34
,35,36,37,38,39,40,41,42,43,44,45,46,47,48,49,50,51,52,53,54,55,56,57,58,59,60,61,62,63,64,6
5,66,67,68,69,70,71,72,73,74,75,76,77,78,79,80,81,82,83,84,85,86,87,88,89,90,91,92,93,94,95,
96,97,98,99,100
      ]
    },
    {
      "field_name": "word_count",
      "field": [ 1000,2000,3000,4000,5000,6000,7000,8000,9000,10000,11000,12000,13000,14000,
15000,16000,17000,18000,19000,20000,21000,22000,23000,24000,25000,26000,27000,28000,29000,30
000,31000,32000,33000,34000,35000,36000,37000,38000,39000,40000,41000,42000,43000,44000,4500
0,46000,47000,48000,49000,50000,51000,52000,53000,54000,55000,56000,57000,58000,59000,60000,
61000,62000,63000,64000,65000,66000,67000,68000,69000,70000,71000,72000,73000,74000,75000,76
000,77000,78000,79000,80000,81000,82000,83000,84000,85000,86000,87000,88000,89000,90000,9100
0,89020,93000,94000,95000,96000,97000,98000,99000,100000
      ]
    },
    {
      "field_name": "book_intro",
      "field": [
[1,1],[2,1],[3,1],[4,1],[5,1],[6,1],[7,1],[8,1],[9,1],[10,1],[11,1],[12,1],[13,1],[14,1],[15
,1],[16,1],[17,1],[18,1],[19,1],[20,1],[21,1],[22,1],[23,1],[24,1],[25,1],[26,1],[27,1],[28,
1],[29,1],[30,1],[31,1],[32,1],[33,1],[34,1],[35,1],[36,1],[37,1],[38,1],[39,1],[40,1],[41,1
],[42,1],[43,1],[44,1],[45,1],[46,1],[47,1],[48,1],[49,1],[50,1],[51,1],[52,1],[53,1],[54,1]
,[55,1],[56,1],[57,1],[58,1],[59,1],[60,1],[61,1],[62,1],[63,1],[64,1],[65,1],[66,1],[67,1],
[68,1],[69,1],[70,1],[71,1],[72,1],[73,1],[74,1],[75,1],[76,1],[77,1],[78,1],[79,1],[80,1],[
```

```
81,1],[82,1],[83,1],[84,1],[85,1],[86,1],[87,1],[88,1],[89,1],[90,1],[91,1],[92,1],[93,1],[94,1],[95,1],[96,1],[97,1],[98,1],[99,1],[100,1]
      ]
    }
  ],
  "num_rows": 100,
  "op_type": "insert"
}';
```

返回结果：

```
{
  "succ_index" : [0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,39,40,41,42,43,44,45,46,47,48,49,50,51,52,53,54,55,56,57,58,59,60,61,62,63,64,65,66,67,68,69,70,71,72,73,74,75,76,77,78,79,80,81,82,83,84,85,86,87,88,89,90,91,92,93,94,95,96,97,98,99],
  "insert_cnt" : 100
}
```

参数说明：

参数	描述	是否必选
table	表名，本例填 book	是
database_name	表所在数据库名	否，默认 default
fields_data	待插入数据	是
field_name	列名	是
field	列数据	是
num_rows	待插入行数，需要等于各 field 数组长度	是
op_type	操作类型	否，默认为 insert

图表 31 插入操作参数说明

5.3.2. 删除

本节介绍如何从 Hippo 表删除数据。Hippo 会返回被成功删除的数据的下标以及总共被删除的数据总条数，如果出现行级错误(比如主键不存在)，Hippo 会返回具体的行级错误。

Restful API：

```
curl -u shiva:shiva -XPUT 'localhost:8902/hippo/v1/{table}/_bulk?pretty' -H 'Content-Type: application/json' -d'{
  "fields_data": [
    {
      "field_name": "book_id",
      "field": [
```

```
1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34
,35,36,37,38,39,40,41,42,43,44,45,46,47,48,49,50,51,52,53,54,55,56,57,58,59,60,61,62,63,64,6
5,66,67,68,69,70,71,72,73,74,75,76,77,78,79,80,81,82,83,84,85,86,87,88,89,90,91,92,93,94,95,
96,97,98,99,100
      ]
    }
  ],
  "num_rows": 100,
  "op_type": "delete"
}';
```

返回结果：

```
{
  "succ_index" : [0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,
28,29,30,31,32,33,34,35,36,37,38,39,40,41,42,43,44,45,46,47,48,49,50,51,52,53,54,55,56,57,58
,59,60,61,62,63,64,65,66,67,68,69,70,71,72,73,74,75,76,77,78,79,80,81,82,83,84,85,86,87,88,8
9,90,91,92,93,94,95,96,97,98,99],
  "delete_cnt" : 100
}
```

参数说明：

参数	描述	是否必选
table	表名，本例填 book	是
database_name	表所在数据库名	否，默认 default
fields_data	带删除数据	是
filed_name	主键名	是
field	主键值	是
num_rows	待删除数据条数，需要等于各 field 数组长度	是
op_type	操作类型，delete 表示删除	是

图表 32 删除操作参数说明

5.3.3. 更新

本节介绍如何更新 Hippo 中的数据。Hippo 会返回更新成功数据的下标以及总共更新成功的数据条数，如果出现行级错误，Hippo 会返回具体的行级错误。

Restful API：

```
curl -u shiva:shiva -XPUT 'localhost:8902/hippo/v1/{table}/_bulk?pretty' -H 'Content-
Type: application/json' -d'{
  "fields_data": [
    {
```

```
      "field_name": "book_id",
      "field": [
1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34
,35,36,37,38,39,40,41,42,43,44,45,46,47,48,49,50,51,52,53,54,55,56,57,58,59,60,61,62,63,64,6
5,66,67,68,69,70,71,72,73,74,75,76,77,78,79,80,81,82,83,84,85,86,87,88,89,90,91,92,93,94,95,
96,97,98,99,100
      ]
    },
    {
      "field_name": "word_count",
      "field": [ 1000,2000,3000,4000,5000,6000,7000,8000,9000,10000,11000,12000,13000,14000,
15000,16000,17000,18000,19000,20000,21000,22000,23000,24000,25000,26000,27000,28000,29000,30
000,31000,32000,33000,34000,35000,36000,37000,38000,39000,40000,41000,42000,43000,44000,4500
0,46000,47000,48000,49000,50000,51000,52000,53000,54000,55000,56000,57000,58000,59000,60000,
61000,62000,63000,64000,65000,66000,67000,68000,69000,70000,71000,72000,73000,74000,75000,76
000,77000,78000,79000,80000,81000,82000,83000,84000,85000,86000,87000,88000,89000,90000,9100
0,89020,93000,94000,95000,96000,97000,98000,99000,100000
      ]
    },
    {
      "field_name": "book_intro",
      "field": [
[1,1],[2,1],[3,1],[4,1],[5,1],[6,1],[7,1],[8,1],[9,1],[10,1],[11,1],[12,1],[13,1],[14,1],[15
,1],[16,1],[17,1],[18,1],[19,1],[20,1],[21,1],[22,1],[23,1],[24,1],[25,1],[26,1],[27,1],[28,
1],[29,1],[30,1],[31,1],[32,1],[33,1],[34,1],[35,1],[36,1],[37,1],[38,1],[39,1],[40,1],[41,1
],[42,1],[43,1],[44,1],[45,1],[46,1],[47,1],[48,1],[49,1],[50,1],[51,1],[52,1],[53,1],[54,1]
,[55,1],[56,1],[57,1],[58,1],[59,1],[60,1],[61,1],[62,1],[63,1],[64,1],[65,1],[66,1],[67,1],
[68,1],[69,1],[70,1],[71,1],[72,1],[73,1],[74,1],[75,1],[76,1],[77,1],[78,1],[79,1],[80,1],[
81,1],[82,1],[83,1],[84,1],[85,1],[86,1],[87,1],[88,1],[89,1],[90,1],[91,1],[92,1],[93,1],[9
4,1],[95,1],[96,1],[97,1],[98,1],[99,1],[100,1]
      ]
    }
  ],
  "num_rows": 100,
  "op_type": "update"
}';
```

返回结果：

```
{
  "succ_index" : [0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,
28,29,30,31,32,33,34,35,36,37,38,39,40,41,42,43,44,45,46,47,48,49,50,51,52,53,54,55,56,57,58
,59,60,61,62,63,64,65,66,67,68,69,70,71,72,73,74,75,76,77,78,79,80,81,82,83,84,85,86,87,88,8
9,90,91,92,93,94,95,96,97,98,99],
  "update_cnt" : 100
```

```
}
```

参数说明：

参数	描述	是否必选
table	表名，本例填 book	是
database_name	表所在数据库名	否，默认 default
fields_data	待更新数据	是
field_name	待更新列名	是
field	列数据	是
num_rows	待更新行数，需要等于各 field 数组长度	是
op_type	操作类型，支持 update 与 upsert	是

图表 33 更新操作参数说明

5.3.4. 拷贝表

Hippo 支持在同集群、跨集群的两张表之间拷贝数据，以下示例假设在本集群已存在一张 schema 相同的 book1 表。

Restful API：

```
curl -u shiva:shiva -XPOST 'localhost:8902/hippo/v1/_copy_by_query?pretty' -H 'Content-
Type: application/json' -d'{
  "source_table" : {
    "database_name" : "default",
    "table_name" : "book",
    "remote_info" : {
      "master_group" : "localhost:8902",
      "username" : "shiva",
      "password" : "shiva"
    }
  },
  "dest_table" : {
    "database_name" : "default",
    "table_name" : "book1"
  },
  "fields" : ["book_id", "word_count", "book_intro"],
  "fields_projection": [
    {
      "book_id" : "book_id1"
    },
    {
      "word_count" : "word_count1"
    },
```

```
    {
      "book_intro" : " book_intro1"
    }
  ],
  "expr" : "word_count >= 11000",
  "op_type" : "insert",
  "wait_for_completion" : true,
  "timeout" : "2m"
}';
```

返回结果：

```
{
  "job_id" : "519aa728acd94c678a6ba600a62da877",
  "job_status" : "SHIVA_JOB_SUCCESS",
  "inserted_number" : 90,
  "failed_number" : 0,
  "task_results" : [
    {
      "id" : "6f776825ddd449d9b37723bc6aa2abbc",
      "status" : "TASK_SUCCESS",
      "server" : "172.29.203.203:27861",
      "inserted_number" : 90,
      "failed_number" : 0,
      "execute_time" : 0.032
    }
  ]
}
```

参数说明：

参数	描述	是否必选
source_table	源表信息	是
database_name(source_table)	源表数据库	否，默认为 default
table_name(source_table)	源表名	是
remote_info	如源表位于其他集群，需要描述远端集群信息	否
master_group(remote_info)	远端集群 Master Group	是
username(remote_info)	用于连接远端集群的用户名	否，默认使用匿名登录
password(remote_info)	用于连接远端集群的密码	否，默认使用匿名登录
dest_table	目标表信息，目标表位于本集群，需要预先创建	是

database_name(dest_table)	目标表数据库	否，默认为 default
table_name(dest_table)	目标表名	是
fields	需要拷贝的列集	否，默认为所有列
fields_projection	列名投影，如源表与目标表列名不一致，可通过列名投影指定映射关系	否，默认使用原表列名
expr	从源表扫描数据的过滤条件	否，默认全量
op_type	写入目标表使用 insert 或 upsert	否，默认 insert
wait_for_completion	是否同步等待任务结束	否，默认 true
timeout	等待时长	否，默认 5 分钟

图表 34 拷贝表操作参数说明

本例可以简单写为：

```
curl -u shiva:shiva -XPOST 'localhost:8902/hippo/v1/_copy_by_query?pretty' -H 'Content-Type: application/json' -d'{
  "source_table" : {
    "table_name" : "book"
  },
  "dest_table" : {
    "table_name" : "book1"
  },
  "expr" : "word_count >= 11000"
}';
```

5.3.5. 批量删除

本节介绍在 Hippo 中如何进行批量删除操作，使用批量删除操作，用户可以方便的以同步或异步的方式，按过滤条件对表中的数据进行删除操作。

Restful API：

```
curl -u shiva:shiva -XPOST 'localhost:8902/hippo/v1/_delete_by_query?pretty' -H 'Content-Type: application/json' -d'{
  "database_name" : "default",
  "table_name" : "book1",
  "expr" : "word_count >= 11000",
  "wait_for_completion" : true,
  "timeout" : "2m"
}';
```

返回结果：

```
{
  "job_id" : "166882d19e1e4274a155a0523828ebc0",
  "job_status" : "SHIVA_JOB_SUCCESS",
  "deleted_number" : 90,
  "failed_number" : 0,
  "task_results" : [
    {
      "id" : "5cf0fa8728e245969071b28e94bc143a",
      "status" : "TASK_SUCCESS",
      "server" : "192.168.0.110:27851",
      "deleted_number" : 90,
      "failed_number" : 0,
      "execute_time" : 0.004
    }
  ]
}
```

参数说明：

参数	描述	是否必选
database_name	表数据库	否，默认为 default
table_name	表名	是
expr	过滤条件	否，默认全量
wait_for_completion	是否同步等待任务结束	否，默认 true
timeout	等待时长	否，默认 5 分钟

图表 35 批量删除操作参数说明

5.4. 索引类操作

5.4.1. 创建向量索引

本节介绍如何创建向量索引。Hippo 中的向量索引用于进行向量相似性搜索，如果未创建向量索引，Hippo 将无法进行向量相似性搜索。以下示例将创建一个聚类中心数量为 10 的 IVF_FLAT 索引，使用欧式距离(L2)作为相识度度量。

Restful API：

```
curl -u shiva:shiva -XPUT 'localhost:8902/hippo/v1/{table}/_create_embedding_index?pretty' -H 'Content-Type: application/json' -d'{
  "field_name" : "book_intro",
  "index_name" : "ivf_flat_index",
```

```
  "metric_type" : "l2",
  "index_type": "IVF_FLAT",
  "params": {
    "nlist" : 10
  }
}';
```

返回结果：

```
{
  "acknowledged" : true
}
```

参数说明：

参数	描述	选项
table	表名，本例填 book	
database_name	表所在数据库名	
field_name	需要创建向量索引的向量列	
index_name	向量索引名	
metric_type	向量相识度度量类型	• L2(欧式距离，Euclidean distance) • IP(点积，Inner product)
index_type	向量索引类型	• FLAT • IVF_FLAT • IVF_SQ • IVF_PQ • IVF_PQ_FS • HNSW
params	向量索引参数，与具体向量索引类型相关	

图表 36 创建向量索引操作参数说明

5.4.2. 激活向量索引

当前在创建向量索引后，Hippo 并不会自动激活向量索引，这是因为在表存量数据量较大时，向量索引的构建将非常耗费资源，用户可以选择在合适的时间激活向量索引，一旦向量索引被激活，Hippo 将自动维护索引，后续对表数据的增、删、改都将同步更新向量索引。本小节介绍如何激活向量索引。

Restful API：

```
curl -u shiva:shiva -
XPOST 'localhost:8902/hippo/v1/{table}/_activate_embedding_index?pretty' -H 'Content-
Type: application/json' -d'{
  "index_name" : "ivf_flat_index",
  "wait_for_completion" : true,
  "timeout" : "2m"
}';
```

返回结果：

```
{
  "job_id" : "2c7fcdd7cd23479bb8660a86703e5c2f",
  "job_status" : "SHIVA_JOB_SUCCESS",
  "embedding_number" : 100,
  "task_results" : [
    {
      "id" : "cecfe25db0b840b899cb5b3806bdcfcc",
      "status" : "TASK_SUCCESS",
      "server" : "172.29.40.26:27861",
      "embedding_number" : 100,
      "execute_time" : 0.233
    }
  ]
}
```

参数说明：

参数	描述	是否必选
table	表名，本例填 book	是
database_name	表所在数据库名	否，默认为 defalt
index_name	向量索引名	是
wait_for_completion	是否同步等待激活完成	是
timeout	等待时间	如 wait_for_completion 为 true，需要设置 timeout 时间

图表 37 激活向量索引操作参数说明

激活向量索引时，Hippo 会扫描存量数据以构建向量索引，在数据量较大时这个过程耗时可能较长，可以将 wait_for_completion 设置为 false 以进行异步激活，次数 Hippo 将返回激活任务的任务 id，用户可以通过任务 id 查看激活任务状态。激活任务完成后，将汇总本次激活操作中用于构建索引的向量数，激活任务不会阻塞表的写入。

5.4.3. 释放向量索引

当前向量索引为全内存，Hippo 支持释放向量索引，以降低内存开销，被释放的向量索引将不再占用内存，但也无法提供向量搜索操作。本节介绍释放向量索引操作。

Restful API：

```
curl -u shiva:shiva -
XPOST 'localhost:8902/hippo/v1/{table}/_release_embedding_index?pretty' -H 'Content-
Type: application/json' -d'{
  "index_name" : "ivf_flat_index",
  "wait_for_completion" : true,
  "timeout" : "2m"
```

```
}';
```

返回结果：

```
{
  "job_id" : "3d585076c8994ef881eab2f16daaf096",
  "job_status" : "SHIVA_JOB_SUCCESS",
  "embedding_number" : 100,
  "task_results" : [
    {
      "id" : "0c4502bcefd4436a9efdddf78902c7a7",
      "status" : "TASK_SUCCESS",
      "server" : "172.29.40.26:27861",
      "embedding_number" : 100,
      "execute_time" : 0.014
    }
  ]
}
```

参数说明：

释放向量索引的参数含义与激活向量索引相同，类似激活任务，释放任务完成后，将汇总本次释放的向量数，释放任务不会阻塞表的写入。

5.4.4. 加载向量索引

在使用向量索引之前，需要确保向量索引被加载到内存中。本节介绍加载向量索引操作。

Restful API：

```
curl -u shiva:shiva -XPOST 'localhost:8902/hippo/v1/{table}/_load_embedding_index?pretty' -
H 'Content-Type: application/json' -d'{
  "index_name" : "ivf_flat_index",
  "wait_for_completion" : true,
  "timeout" : "2m"
}';
```

返回结果：

```
{
  "job_id" : "b0dc39c6023041228a5e2888174f6396",
  "job_status" : "SHIVA_JOB_SUCCESS",
  "embedding_number" : 100,
  "task_results" : [
    {
      "id" : "ee4db1d2c5ee455db85db072b90b5e4a",
      "status" : "TASK_SUCCESS",
      "server" : "172.29.40.26:27861",
```

```
      "embedding_number" : 100,
      "execute_time" : 0.014
    }
  ]
}
```

参数说明：
加载向量索引的参数含义与激活向量索引相同，类似激活任务，加载任务完成后，将汇总本次加载的向量数，加载任务不会阻塞表的写入。

5.4.5. 查看向量索引数据

用户可以查看一张表向量索引的详细数据信息。Hippo 会返回每个分片上向量索引所有的段信息。

Restful API：

```
curl -u shiva:shiva -XGET 'localhost:8902/hippo/v1/{table}/_get_embedding_index?pretty'
```

返回结果：

```
{
  "default#book" : {
    "table_id" : "361165c970d1470886c435a86e3eae2f",
    "shards" : [
      {
        "tablet_id" : 0,
        "address" : "172.29.40.26:27851",
        "indexes" : [
          {
            "index_id" : 0,
            "state" : "PUBLIC",
            "flat_segments" : [
              {
                "min_id" : -1,
                "max_id" : -1,
                "embedding_num" : 0,
                "deleted_embedding_num" : 0
              },
              {
                "min_id" : 0,
                "max_id" : 99,
                "embedding_num" : 100,
                "deleted_embedding_num" : 0
              }
            ]
          }
```

```
        ]
      }
    ]
  }
}
```

参数说明：

参数	描述	是否必选
table	表名，本例填 book	是
database_name	表所在数据库名	否，默认为 defalt

图表 38 查看向量索引数据操作参数说明

5.4.6. 删除向量索引

本节描述如何删除一个向量索引。

Restful API：

```
curl -u shiva:shiva -
XDELETE 'localhost:8902/hippo/v1/{table}/_drop_embedding_index?pretty' -H 'Content-
Type: application/json' -d'{
  "index_name" : "book_intro_index"
}';
```

返回结果：

```
{
  "acknowledged" : true
}
```

参数说明：

参数	描述	是否必选
table	表名，本例填 book	是
database_name	表所在数据库名	否，默认为 defalt
index_name	要删除的向量索引名，如需删除多个索引，使用 "index_name" : ["a","b","c"]	是

图表 39 删除向量索引操作参数说明

5.4.7. 创建标量索引

在 Hippo 中非向量类型被成为标量，与传统数据库类似，用户可以通过创建标量索引加速对标量类型的操作。本节介绍如何创建标量索引。与向量索引不同，标量索引在创建后会被自动激活。

Restful API：

```
curl -u shiva:shiva -XPUT 'localhost:8902/hippo/v1/{table}/_create_scalar_index?pretty' -
H 'Content-Type: application/json' -d'{
  "index_name" : "index",
  "field_names" : ["word_count"]
}';
```

返回结果：

```
{
  "acknowledged" : true
}
```

参数说明：

参数	描述	是否必选
table	表名，本例填 book	是
database_name	表所在数据库名	否，默认为 defalt
index_name	索引名	是
field_names	创建索引的字段名，Hippo 支持单列索引与组合索引	是

图表 40 创建标量索引操作参数说明

5.4.8. 删除标量索引

本节介绍如何删除标量索引。

Restful API：

```
curl -u shiva:shiva -XDELETE 'localhost:8902/hippo/v1/{table}/_drop_scalar_index?pretty' -
H 'Content-Type: application/json' -d'{
  "index_name" : "index"
}';
```

返回结果：

```
{
  "acknowledged" : true
}
```

参数说明：

参数	描述	是否必选
table	表名，本例填 book	是
database_name	表所在数据库名	否，默认为 defalt
index_name	要删除的向量索引名，如需删除多个索引，使用"index_name" : ["a","b","c"]	是

图表 41 删除标量索引操作参数说明

5.5. 查询类操作

5.5.1. 向量相似性检索

本节描述如何进行向量相似度搜索。Hippo 中的向量相似性搜索计算查询向量与表中向量的距离，返回最相似的结果集。通过指定标量过滤条件，用户可以进行向量与标量的混合搜索。有关向量索引，向量检索等更详细的内容，请参考下方第 6 章节。

Restful API：

```
curl -u shiva:shiva -XGET 'localhost:8902/hippo/v1/{table}/_search?pretty' -H 'Content-Type: application/json' -d'{
  "output_fields": ["book_id"],
  "search_params": {
    "anns_field": "book_intro",
    "topk": 2,
    "params": {
      "nprobe": 10
    },
    "embedding_index": "ivf_flat_index"
  },
  "vectors": [ [0.1,0.2], [0.3, 0.4] ]
}';
```

返回结果：

```
{
  "num_queries" : 2,
  "top_k" : 2,
  "results" : [
    {
      "query" : 0,
      "fields_data" : [
        {
          "field_name" : "book_id",
          "field_values" : [1,2],
          "scores" : [1.45,4.25]
        }
      ]
    },
    {
      "query" : 1,
      "fields_data" : [
```

```
        {
          "field_name" : "book_id",
          "field_values" : [1,2],
          "scores" : [0.85,3.25]
        }
      ]
    }
  ]
}
```

参数说明：

参数	描述	是否必选
table	表名，本例填 book	是
database_name	表所在数据库名	否，默认为 defalt
output_fields	搜索结果返回的列	是
anns_field	搜索的向量列名	是
topk	搜索返回结果数	是
params	向量索引相关搜索参数	否
embedding_index	本次搜索使用的向量索引	否，默认使用第一个已激活且对 anns_field 创建的向量索引
vectors	搜索向量	是

图表 42 向量相似性搜索操作参数说明

5.5.2. 向量标量混合搜索

本节介绍如何进行向量标量混合搜索。混合搜索本质上是为向量搜索指定了一系列过滤条件，Hippo 将只会返回满足过滤条件的搜索结果。创建标量索引将有助于提升混合搜索的性能与准确率。

Restful API：

```
curl -u shiva:shiva -XGET 'localhost:8902/hippo/v1/{table}/_search?pretty' -H 'Content-Type: application/json' -d'{
  "output_fields": ["book_id"],
  "search_params": {
    "anns_field": "book_intro",
    "topk": 2,
    "params": {
      "k_factor" : 100
    }
  },
  "vectors": [ [0.1,0.2], [0.3, 0.4] ],
  "dsl": "word_count >= 11000"
}';
```

返回结果：


```
{
  "num_queries" : 2,
  "top_k" : 2,
  "results" : [
    {
      "query" : 0,
      "fields_data" : [
        {
          "field_name" : "book_id",
          "field_values" : [11,12],
          "scores" : [119.44999,142.24998]
        }
      ]
    },
    {
      "query" : 1,
      "fields_data" : [
        {
          "field_name" : "book_id",
          "field_values" : [11,12],
          "scores" : [114.85,137.25]
        }
      ]
    }
  ]
}
```


参数说明：

参数	描述	是否必选
table	表名，本例填 book	是
database_name	表所在数据库名	否，默认为 defalt
output_fields	搜索结果返回的列	是
anns_field	搜索的向量列名	是
topk	搜索返回结果数	是
params	向量索引相关搜索参数	否
embedding_index	本次搜索使用的向量索引	否，默认使用第一个已激活且对 anns_field 创建的向量索引

vectors	搜索向量	是
dsl	标量过滤条件	是

图表 43 向量标量混合搜索操作参数说明

5.5.3. 标量查询

本节介绍如何通过设置标量过滤条件查询数据。

Restful API：

```
curl -u shiva:shiva -XGET 'localhost:8902/hippo/v1/{table}/_query?pretty' -H 'Content-
Type: application/json' -d'{
  "output_fields": ["book_id", "book_intro"],
  "expr" : "word_count >= 11000",
  "limit" : 2
}';
```

返回结果：

```
{
  "fields_data" : [
    {
      "field_name" : "book_id",
      "field_values" : [11,12]
    },
    {
      "field_name" : "book_intro",
      "field_values" : [[11.0,1.0],[12.0,1.0]]
    }
  ]
}
```

参数说明：

参数	描述	是否必选
table	表名，本例填 book	是
database_name	表所在数据库名	否，默认为 defalt
output_fields	查询结果返回的列	否
limit	查询返回结果数	否
expr	过滤条件	否

图表 44 标量查询操作参数说明

5.6. 用户 ACL 相关

本节介绍 Hippo 中的用户 ACL 相关操作。

5.6.1. 创建用户

本节描述如何在 Hippo 中创建用户。

Restful API：

```
curl -u shiva:shiva -XPUT localhost:8902/hippo/v1/_security/user/{user_name}?pretty -
H 'Content-Type: application/json' -d '{
  "password" : "<password>",
  "metadata" : {
    "is_super" : false,
    "can_create_role": false,
    "can_create_db": false
  }
}';
```

返回结果：

```
{
  "user" : {
    "created" : true
  },
  "created" : true
}
```

参数说明：

参数	描述	选项
{user_name}	新建用户名	
password	用户密码	
is_super	是否为超级用户	true 或 false
can_create_role	是否可执行新建用户操作	true 或 false
can_create_db	是否可以执行创建数据库操作	true 或 false

图表 45 创建用户操作参数说明

5.6.2. 查看用户

本节描述如何查看 Hippo 中当前存在的用户。

Restful API：

```
curl -u shiva:shiva -XGET localhost:8902/hippo/v1/_security/user/{user_name}?pretty
```

返回结果：

```
{
  "public" : {
    "id" : "00000000000000000000000000000000",
    "is super" : false,
    "permit to create role" : false,
    "permit to create db" : false,
    "cant login" : false,
    "create date time" : "2023-05-22 17:05:37"
  },
  "shiva" : {
    "id" : "5f08847d6bc44c7daf9928b4ebe9443c",
    "is super" : true,
    "permit to create role" : true,
    "permit to create db" : true,
    "cant login" : true,
    "create date time" : "2023-05-22 17:05:37"
  },
  "melon" : {
    "id" : "c95316b8414e4821b235173560bc85c5",
    "is super" : false,
    "permit to create role" : false,
    "permit to create db" : false,
    "cant login" : true,
    "create date time" : "2023-05-22 17:07:28"
  }
}
```

参数说明：

参数	描述	是否必选
user_name	支持*通配	否，默认为*，返回所有用户

图表 46 查看用户操作参数说明

5.6.3. 删除用户

本节描述如何删除 Hippo 用户。

Restful API：

```
curl -u shiva:shiva -XDELETE localhost:8902/hippo/v1/_security/user/{user_name}?pretty
```

返回结果：

```
{
  "found" : true
}
```

参数说明：

参数	描述	是否必选
user_name	待删除用户名	是

图表 47 删除用户操作参数说明

5.6.4. 修改用户

本节描述如何修改用户属性。

Restful API：

```
curl -u shiva:shiva -
XPOST localhost:8902/hippo/v1/_security/user/{user_name}/_alter?pretty -H 'Content-
Type: application/json' -d '{
  "password" : "<password>",
  "metadata" : {
    "is_super" : false,
    "can_create_role": false,
    "can_create_db": false
  }
}';
```

返回结果：

```
{
  "acknowledged" : true
}
```

参数说明：
修改用户对应的参数同创建用户操作。

5.6.5. 赋权

本节描述如何对用户进行赋权。

Restful API：

```
curl -u shiva:shiva -XPOST "localhost:8902/hippo/v1/_security/acl/{user_name}?pretty" -
H 'Content-Type: application/json' -d'
{
  "table_name": "book",
  "database_name" : "default",
  "privileges": ["all"]
}';
```

返回结果：

```
{
  "acknowledged" : true
}
```

参数说明：

参数	描述	是否必选
user_name	待删除用户名	是
table_name	待赋权表名	否，如未指定，表示对数据库赋权
database_name	待赋权表所属 database	否，默认 default
privileges	具体权限，"create", "access", "all"	是

图表 48 赋权操作参数说明

如指定 table_name，表示对表进行赋权，如未指定，表示对 database 进行赋权，此时 database_name 必须指定。

5.6.6. 删除权限

本节描述如何删除用户权限。

Restful API：

```
curl -u shiva:shiva -XDELETE "localhost:8902/hippo/v1/_security/acl/{user_name}?pretty" -
H 'Content-Type: application/json' -d'
{
  "table_name": "book",
  "privileges": ["access"]
}';
```

返回结果：

```
{
  "acknowledged" : true
}
```

参数说明：

删除用户权限相关操作涉及的权限同上一节赋权操作。

5.6.7. 查看用户权限

本节描述如何查看用户权限。

Restful API：

```
curl -u shiva:shiva -
XGET "localhost:8902/hippo/v1/_security/user/_privileges/{username}?pretty"
```

返回结果：

```
{
  "default#book" : {
    "id" : "095a7024526c43f59c97994803906944",
    "owner" : "melon",
    "acl" : [
      {
        "role" : "melon",
        "grantor" : "shiva",
        "privileges" : [
          {
            "privilege" : "PRIV_ALL",
            "with grant options" : true
          }
        ]
      }
    ]
  }
}
```

参数说明：

参数	描述	是否必选
user_name	待删除用户名	是

图表 49 查看用户权限操作参数说明

5.6.8. 查看表权限

本节描述如何查看表权限。

Restful API：

```
curl -u shiva:shiva -XGET "localhost:8902/hippo/v1/_security/tables/{table}?pretty"
```

返回结果：

```
{
```

```
  "book" : {
    "id" : "fc6feff4f303455a9347f9aab323dfc8",
    "owner" : "shiva",
    "acl" : [
      {
        "role" : "melon",
        "grantor" : "shiva",
        "privileges" : [
          {
            "privilege" : "PRIV_ALL",
            "with grant options" : true
          }
        ]
      }
    ]
  }
}
```

参数说明：

参数	描述	是否必选
table	表名	否，支持通配，默认为*

图表 50 查看表权限操作参数说明

5.7. 任务相关

在 Hippo 中，比较耗时的操作如激活、加载向量索引实际上是一个分布式任务，用户可以通过任务相关接口查看、删除任务。

5.7.1. 查看任务

Restful API：

```
curl -u shiva:shiva -XGET "localhost:8902/hippo/v1/_jobs?pretty" -H 'Content-
Type: application/json' -d'
{
  "job_ids" : ["fc6feff4f303455a9347f9aab323dfc8"],
  "action_patterns" : ["hippo*"]
}';
```

返回结果：

```
{
  "jobs" : [
    {
      "job_id" : "810935a1d91a46b7af2ec35013454fed",
```

```
      "job_status" : "SHIVA_JOB_SUCCESS",
      "embedding_number" : 100,
      "task_results" : [
        {
          "id" : "54ab52493dfb4bab9fb7742d850c64c4",
          "status" : "TASK_SUCCESS",
          "server" : "172.29.40.26:27841",
          "embedding_number" : 100,
          "execute_time" : 0.248
        }
      ]
    }
  ]
}
```

参数说明：

参数	描述	是否可选
job_ids	任务 id，支持同时查看多个任务	是
action_patterns	任务 action 名，支持通配	是

图表 51 查看任务操作参数说明

如同时指定 job_ids 与 action_patterns，优先使用 job_ids。

5.7.2. 删除任务

当删除一个未完成任务时，该任务会首先被停止。

Restful API：

```
curl -u shiva:shiva -XDELETE "localhost:8902/hippo/v1/_jobs/{jod_id}?pretty"
```

返回结果：

```
{
  "acknowledged" : true
}
```

参数说明：

参数	描述	是否可选
job_id	待删除任务 id	否

图表 52 删除任务操作参数说明

5.8. 回收站操作

当一张 Hippo 表被删除时，表数据会被保留在回收站中，直到一段时候后自动删除，用户可以查看、恢复或删除回收站中的表。

5.8.1. 查看回收站中的表

Restful API：

```
curl -u shiva:shiva -XGET 'localhost:8902/hippo/v1/_cat/trash?database_name=book&v'
```

返回结果：

```
epoch      timestamp table uuid                             deletion_date_time
1687918466 02:14:26  book1 6ff476b55ca14b4784fdbfe89225d6e3 2023-06-28 10:13:41
```

5.8.2. 删除回收站中的表

Restful API：

```
curl -u shiva:shiva -
XDELETE 'localhost:8902/hippo/v1/_trash/{table}?database_name=book&pretty'
```

返回结果：

```
{
  "acknowledged" : true
}
```

参数说明：

参数	描述	是否必选
table	表示待匹配的表名，支持以*通配，支持指定多个 pattern，多个 pattern 以逗号分割	否，默认为*，表示删除本数据库回收站里所有表
database_name	database 名	否，默认为 default

图表 53 删除回收站中的表操作参数说明

6. 向量检索

向量检索就是在一个给定向量数据集中，按照某种度量方式，检索出与查询向量相近的 K 个向量 (K-Nearest Neighbor，KNN)，但由于 KNN 计算量过大，我们通常只关注近似近邻 (Approximate Nearest Neighbor，ANN) 问题。

6.1. 向量度量

度量指标用于评估两个向量的相似程度，Hippo 支持的度量指标如下：

度量指标	索引类型
• 欧式距离 Euclidean distance (L2) • 内积 Inner product (IP)	• FLAT • IVF_FLAT • IVF_SQ • IVF_PQ • IVF_PQ_FS • HNSW

图表 54 向量度量指标

6.1.1. 欧氏距离

欧式距离也称欧几里得距离，是最常见的距离度量，衡量的是多维空间中两个点之间的绝对距离。

1. 在二维和三维空间中的欧式距离的就是两点之间的距离，二维的公式是：

$$d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

2. 三维公式是：

$$d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2 + (z_1 - z_2)^2}$$

3. 推广到 N 维空间，欧式距离的公式是：

$$d(x,y) = d(y,x) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$$

6.1.2. 内积

两个向量之间的内积距离定义如下：

$$p(X,Y) = X \cdot Y = \sum_{i=1}^{n} x_i \times y_i$$

如果需要度量向量的方向而不是的大小，IP 会更有用。注意，如果使用内积作为度量指标，需要首先对向量进行归一化，经过归一化后，内积相似度等于余弦相似度。

6.2. 向量索引

本节介绍 Hippo 支持的向量索引。Hippo 支持为同一向量字段同时创建多个向量索引。Hippo 支持以下向量索引。

索引	类型	适用场景
FLAT		• 适用于小数据集 • 召回率 100%
IVF_FLAT	Quantization-based Index	• 查询性能较高 • 召回率高
IVF_SQ	Quantization-based Index	• 查询性能较高

		• 内存开销低 • 召回率较高
IVF_PQ	Quantization-based Index	• 查询性能高 • 内存开销低 • 召回率低于 IVF_SQ
IVF_PQ_FS	Quantization-based Index	• 查询性能高 • 内存开销低 • 召回率低于 IVF_SQ
HNSW	Graph-based Index	• 查询性能高 • 召回率高 • 内存开销大

图表 55 向量索引简介

6.2.1. FLAT

FLAT 是唯一一种能够保证 100%召回率的向量索引，FLAT 索引不会对向量进行压缩，采用暴力搜索进行相似度搜索，所以也是最慢的向量索引。FLAT 索引适用于需要 100%召回率的小数据集。

6.2.2. IVF_FLAT

IVF_FLAT 是一种粗量化索引，构建索引时将向量数据集根据相似度聚类到 nlist 个中心点，搜索时仅需考虑距离搜索向量最近的 nprobe 个聚类。nprobe 参数的选择涉及召回率与搜索效率的取舍，nprobe 越大，召回率越高，但效率越低。

构建参数：

参数	**描述**	**范围**
nlist	聚类中心点数	[1,65536]
nprobe	默认搜索中心点数	[1,nlist]

图表 56 IVF_FLAT 构建参数

搜索参数：

参数	**描述**	**范围**
nprobe	搜索中心点数	[1,nlist]

图表 57 IVF_FLAT 搜索参数

6.2.3. IVF_SQ

IVF_FLAT 索引不会进行向量压缩，索引内存占用与原始向量数据一致。当内存受限时，可以考虑使用 IVF_SQ 索引。IVF_SQ8 通过将标量量化技术，将 Float (4bytes) 压缩成 UINT8 (1 byte)，能够有效降低索引的内存占用。

构建参数：

参数	描述	范围
nlist	聚类中心点数	[1,65536]
nprobe	默认搜索中心点数	[1,nlist]

图表 58 IVF_SQ 构建参数

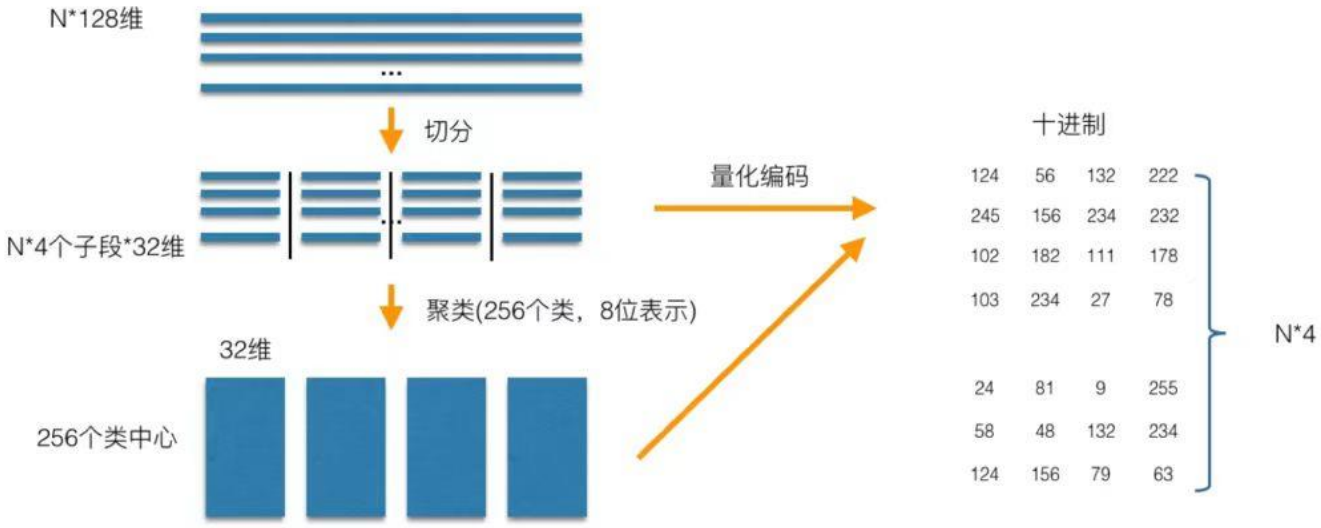
搜索参数：

参数	描述	范围
nprobe	搜索中心点数	[1,nlist]

图表 59 IVF_SQ 搜索参数

6.2.4. IVF_PQ

乘积量化 (Product Quantization，PQ) 是 Herve Jegou 在 2011 年提出的一种非常经典实用的矢量量化索引方法，PQ 将原始高维向量空间统一分解为 M 个低维向量空间的笛卡尔积，然后对分解后的低维向量空间进行量化。乘积量化不再计算目标向量与所有单元中心的距离，而是计算目标向量与各个低维空间的聚类中心的距离，大大降低了算法的时间复杂度和空间复杂度。



图表 60 PQ 乘积量化生成码本和量化过程

IVF_PQ 索引在乘积量化之前，增加了一个 IVF 粗量化过程。IVF_PQ 索引的内存占用与 IVF_SQ 索引更小，但也引入了搜索精度的丢失。

构建参数：

参数	描述	范围
nlist	聚类中心点数	[1,65536]

m	PQ 过程分解系数	需要能被向量维度整除
nbits	存储低维向量使用的 bit 数，低维空间聚类中心点数为 2^nbits	[1,16]，默认为 8
nprobe	默认搜索中心点数	[1,nlist]

图表 61 IVF_PQ 构建参数

搜索参数：

参数	**描述**	**范围**
nprobe	搜索中心点数	[1,nlist]

图表 62 IVF_PQ 搜索参数

6.2.5. IVF_PQ_FS

IVF_PQ_FS 为 IVF_PQ 的 fast scan 版本，使用 simd 加速低维子空间搜索，目前约束 nbits 为 4。

参数	**描述**	**范围**
nlist	聚类中心点数	[1,65536]
m	PQ 过程分解系数	需要能被向量维度整除
nprobe	默认搜索中心点数	[1,nlist]

图表 63 IVF_PQ_FS 构建参数

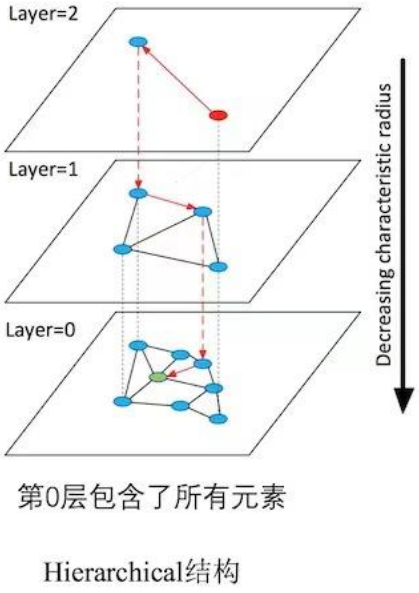
搜索参数：

参数	**描述**	**范围**
nprobe	搜索中心点数	[1,nlist]

图表 64 IVF_PQ_FS 搜索参数

6.2.6. HNSW

HNSW (Hierarchical Navigable Small World Graph) 是一种基于图的索引算法。是 Yury A. Malkov 提出的一种基于图索引的方法，它是在他本人之前工作 上一种改进，通过采用层状结构，将边按特征半径进行分层，使每个顶点在所有层中平均度数变为常数，从而将 NSW 的计算复杂度由多重对数 (Polylogarithmic) 复杂度降到了对数 (logarithmic) 复杂度。



每一个插入的元素对应的最大layer由指数衰减概率分布函数给出

```
mult_ = 1 / log(1.0 * M_);
revSize_ = 1.0 / mult_;
std::uniform_real_distribution<double> distribution(0.0, 1.0);
double r = -log(distribution(level_generator_)) * reverse_size;
```

1) 从最大层layer开始贪心遍历，寻找局部最小值(最近邻)

2) 将当前层找到的最近邻作为进入点(entry point)，在下一层寻找局部最小值

图表 65 HNSW 算法

搜索过程从若干输入点 (随机选取或分割算法) 开始迭代遍历整个近邻图。整个搜索过程可以类比在地图上寻找某个位置的过程：我们可以地球当做最顶层，五大洲作为第二层，国家作为第三层，省份作为第四层……，现在如果要找海淀五道口，我们可以通过顶层以逐步递减的特性半径对其进行路由(第一层地球->第二层亚洲->第三层中国->第四层北京->海淀区)，到了第 0 层后，再在局部区域做更精细的搜索。

构建参数：

参数	**描述**	**范围**
M	每个向量在图中的邻居数	[4,96]
ef_construction	构建图时，每个顶点候选的最近邻居	[8,512]
ef_search	搜索时，默认每个顶点候选的最近邻居数	[top_k,32768]

图表 66 HNSW 构建参数

搜索参数：

参数	**描述**	**范围**
ef_search	搜索时，每个顶点候选的最近邻居数	[top_k,32768]

图表 67 HNSW 搜索参数

7. 性能分析

本节测试主要描述了 Hippo 1.0 在关键测试上的一些性能表现，该份测试同样也是 Hippo 的基准测试，后续版本发布也会在不同版本上进行该测试进行对比分析。

7.1. 术语

Term	Description
nq	一次搜索请求中搜索的向量个数
topk	一次请求中对于要检索的每个向量(依赖 nq)，所能检索到最近距离的向量个数
RT	一次请求从发起到接受响应的时间]
QPS	请求在每秒内成功执行的次数
dataset	测试所用数据集，不同数据集表示不同的业务场景

图表 68 Hippo 性能测试术语

7.2. 测试集群配置

7.2.1. 硬件配置

Hardware	Specification
Nodes	3
CPU	Intel(R) Xeon(R) Gold 5218R CPU @ 2.10GHz
Memory	16*\16 GB RDIMM, 3200 MT/s
DISK	NVMe SSD 2T*4
GPU	NONE

图表 69 性能测试硬件配置

7.2.2. 软件配置

Software	Version
Hippo	v1.0
Transwarp Manager	TDH 9.3.0

图表 70 性能测试软件配置

7.3. 测试集

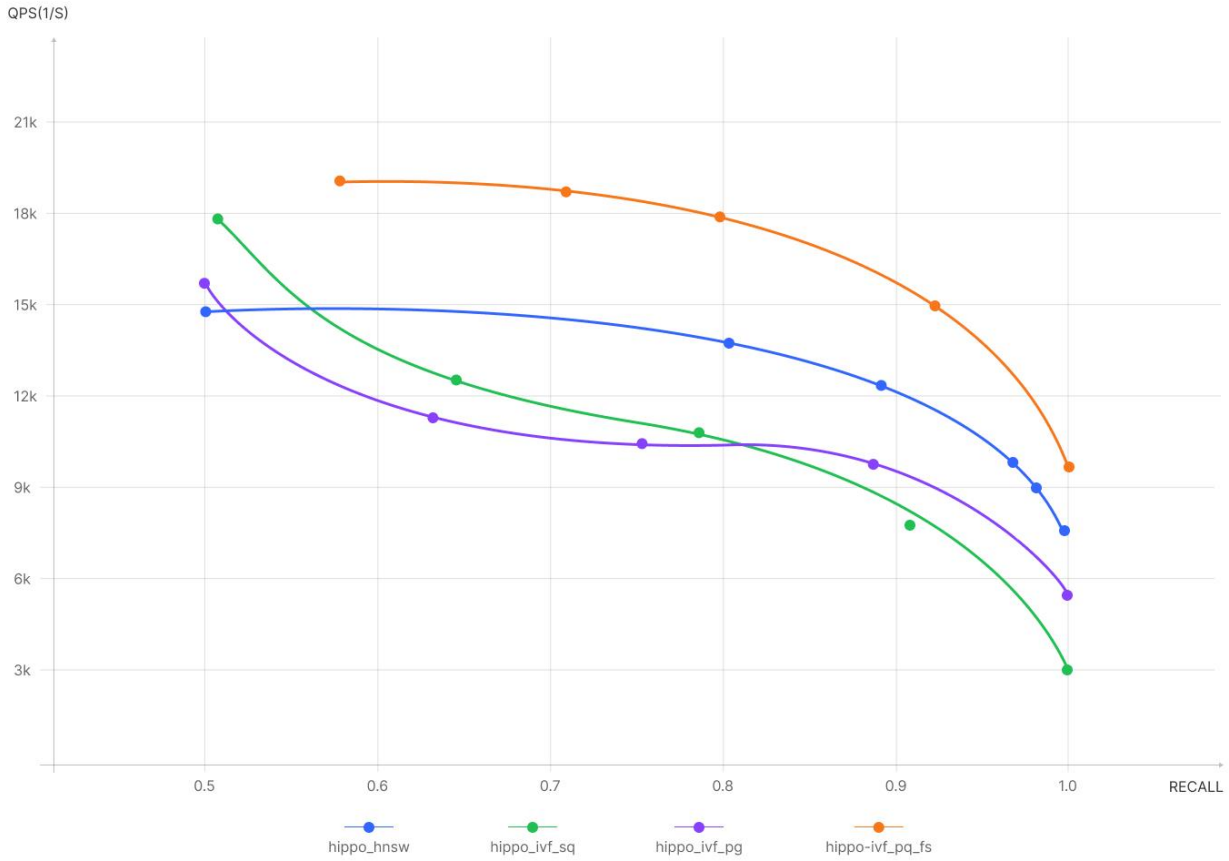
数据集名称	数据集介绍	向量维度	向量总数	查询数量	数据总量	距离类型
Sift-128-euclidean	该数据集是基于 Texmex 的数据集整理，使用 SIFT 算法得到的图片特征向量。	128	1,000,000	10,000	488 MB	L2
Gist-960-euclidean	该数据集是基于 Texmex 的数据集整理，使用 GIST 算法得到的图片特征向量。	960	1,000,000	1,000	3.57 GB	L2
Glove-200-angular	该数据集是互联网文本数据使用 GloVe 算法得到的单词向量。	200	1,183,514	10,000	902 MB	IP
Deep-image-96-angular	该数据集是 ImageNet 图片经过 GoogLeNet 模型训练，从最后一层神经网络提取的向量。	96	9,990,000	10,000	3.57 GB	IP
Glove-100-angular	该数据集是互联网文本数据使用 GloVe 算法得到的单词向量。	100	1,183,514	10,000	463MB	IP
Mnist-784-euclidean	该数据集来自于手写数字识别数据库。	784	60,000	10,000	179 MB	L2

图表 71 性能测试数据集

7.4. 测试场景

由于测试数据较多，为了便于大家在使用过程中能比较轻松的选择相对合适的参数进行使用，本次测试报告会挑出通用场景下不同算法在少量不同参数的具体表现而非全部参数集的表现。

7.4.1. Deep-image-96-angular (k=10)



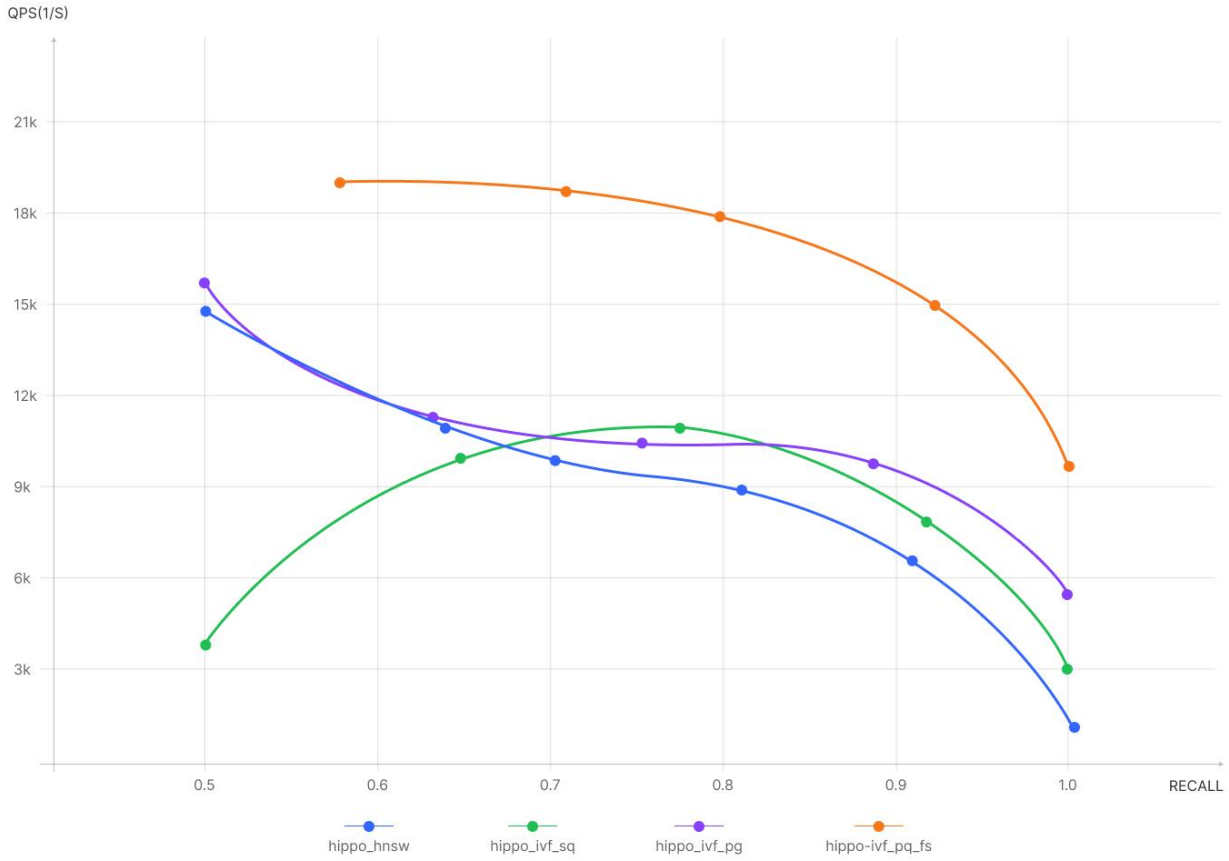
图表 72 Deep-image-96-angular (k=10) 图

ALGORITHM	PARAMETERS	QPS	95th RT	RECALL
hippo_hnsw	Hippo-HNSW({'M': 8, 'efConstruction': 500}, ef_search: 10)	14000	20ms	0.53606
	Hippo-HNSW({'M': 8, 'efConstruction': 500}, ef_search: 40)	13500	20ms	0.80006
	Hippo-HNSW({'M': 64, 'efConstruction': 500}, ef_search: 20)	13000	22ms	0.90231
	Hippo-HNSW({'M': 64, 'efConstruction': 500}, ef_search: 80)	12500	22ms	0.98822
	Hippo-HNSW({'M': 64, 'efConstruction': 500}, ef_search: 400)	9500	29ms	0.99925
	Hippo-HNSW({'M': 64, 'efConstruction': 500}, ef_search: 800)	7000	38ms	0.99955
hippo_ivf_sq	Hippo-IVFSQ-refine({'nlist': 4096, 'sq_type': 'SQfp16'}, nprobe: 10, kfactor: 10)	19000	14ms	0.89522
	Hippo-IVFSQ-refine({'nlist': 1024, 'sq_type': 'SQfp16'}, nprobe: 10, kfactor: 10)	10000	28ms	0.94795
	Hippo-IVFSQ-refine({'nlist': 2048, 'sq_type': 'SQ4'}, nprobe: 100, kfactor: 100)	2900	92ms	0.99737

	Hippo-IVFSQ-refine({'nlist': 4096, 'sq_type': 'SQfp16'}, nprobe: 200, kfactor: 10)	2500	100ms	0.99883
	Hippo-IVFSQ-refine({'nlist': 2048, 'sq_type': 'SQfp16'}, nprobe: 100, kfactor: 100)	3000	100ms	0.99805
hippo_ivf_pq	Hippo-IVFPQ-refine({'nlist': 512, 'M': 24}, nprobe: 10, kfactor: 10)	11000	28ms	0.89889
	Hippo-IVFPQ-refine({'nlist': 512, 'M': 24}, nprobe: 10, kfactor: 1000)	7000	40ms	0.96527
	Hippo-IVFPQ-refine({'nlist': 4096, 'M': 24}, nprobe: 100, kfactor: 10)	10000	31ms	0.95131
	Hippo-IVFPQ-refine({'nlist': 4096, 'M': 24}, nprobe: 100, kfactor: 1000)	7000	42ms	0.99573
hippo-ivf_pq_fs	Hippo-IVFPQFS-refine({'nlist': 1024, 'M': 48}, nprobe: 10, kfactor: 100)	20000	12ms	0.92764
	Hippo-IVFPQFS-refine({'nlist': 2048, 'M': 48}, nprobe: 200, kfactor: 1000)	7600	37ms	0.99946
	Hippo-IVFPQFS-refine({'nlist': 4096, 'M': 24}, nprobe: 50, kfactor: 1000)	14500	19ms	0.97998

图表 73 Deep-image-96-angular (k=10) 表

7.4.2. Gist-960-euclidean (k=10)



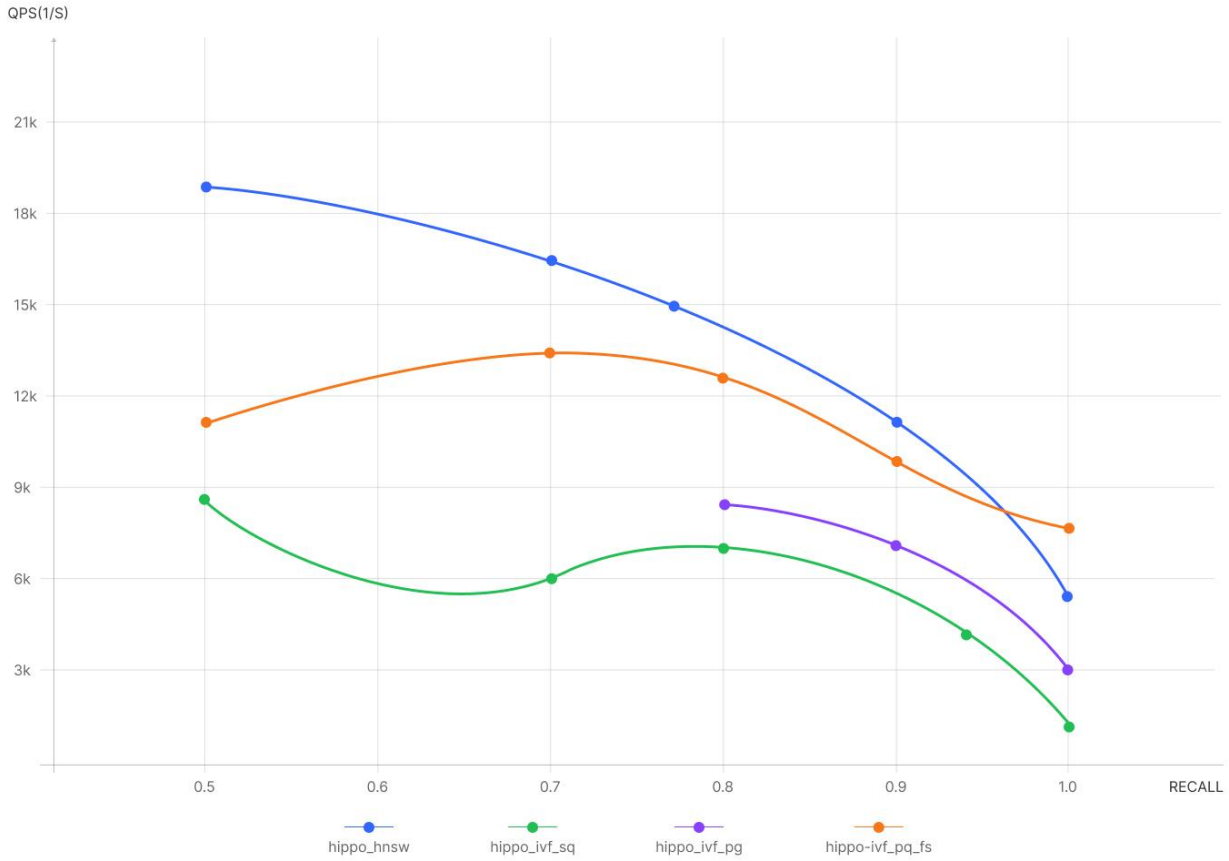
图表 74 Gist-960-euclidean (k=10) 图

ALGORITHM	PARAMETERS	QPS	95th RT	RECALL

hippo_hnsw	Hippo-HNSW({'M': 24, 'efConstruction': 500}, ef_search: 80)	11000	20ms	0.9064
	Hippo-HNSW({'M': 8, 'efConstruction': 500}, ef_search: 800)	6200	50ms	0.9521
	Hippo-HNSW({'M': 48, 'efConstruction': 500}, ef_search: 800)	1800	170ms	0.9983
	Hippo-HNSW({'M': 64, 'efConstruction': 500}, ef_search: 800)	1800	210ms	0.9986
hippo_ivf_pq	Hippo-IVFPQ-refine({'nlist': 4096, 'M': 15}, nprobe: 50, kfactor: 1000)	3300	87ms	0.9083
	Hippo-IVFPQ-refine({'nlist': 2048, 'M': 60}, nprobe: 50, kfactor: 1000)	3500	66ms	0.9468
	Hippo-IVFPQ-refine({'nlist': 1024, 'M': 60}, nprobe: 200, kfactor: 1000)	2800	87ms	0.9996
	Hippo-IVFPQ-refine({'nlist': 4096, 'M': 15}, nprobe: 50, kfactor: 100)	11600	19ms	0.8857
	Hippo-IVFPQ-refine({'nlist': 2048, 'M': 60}, nprobe: 50, kfactor: 100)	11000	21ms	0.9432
	Hippo-IVFPQ-refine({'nlist': 1024, 'M': 60}, nprobe: 200, kfactor: 100)	5500	54ms	0.9932
hippo_ivf_pq_fs	Hippo-IVFPQFS-refine({'nlist': 4096, 'M': 120}, nprobe: 50, kfactor: 100)	11000	19ms	0.8868
	Hippo-IVFPQFS-refine({'nlist': 4096, 'M': 120}, nprobe: 50, kfactor: 1000)	3400	80ms	0.9027
	Hippo-IVFPQFS-refine({'nlist': 4096, 'M': 30}, nprobe: 100, kfactor: 1000)	1100	250ms	0.9535
	Hippo-IVFPQFS-refine({'nlist': 512, 'M': 120}, nprobe: 100, kfactor: 1000)	3300	110ms	0.9969
	Hippo-IVFPQFS-refine({'nlist': 512, 'M': 120}, nprobe: 200, kfactor: 1000)	3100	110ms	0.999
hippo_ivf_sq	Hippo-IVFPQ-refine({'nlist': 2048, 'sq_type': 'SQ8'}, nprobe: 50, kfactor: 10)	4500	62ms	0.9263
	Hippo-IVFPQ-refine({'nlist': 4096, 'sq_type': 'SQ6'}, nprobe: 100, kfactor: 10)	1500	170ms	0.9454
	Hippo-IVFPQ-refine({'nlist': 4096, 'sq_type': 'SQfp16'}, nprobe: 100, kfactor: 10)	3800	83ms	0.9454
	Hippo-IVFPQ-refine({'nlist': 2048, 'sq_type': 'SQfp16'}, nprobe: 100, kfactor: 10)	2200	140ms	0.9779
	Hippo-IVFPQ-refine({'nlist': 1024, 'sq_type': 'SQfp16'}, nprobe: 200, kfactor: 10)	450	780ms	0.9995

图表 75 Gist-960-euclidean (k=10) 表

7.4.3. Glove-100-angular (k=10)



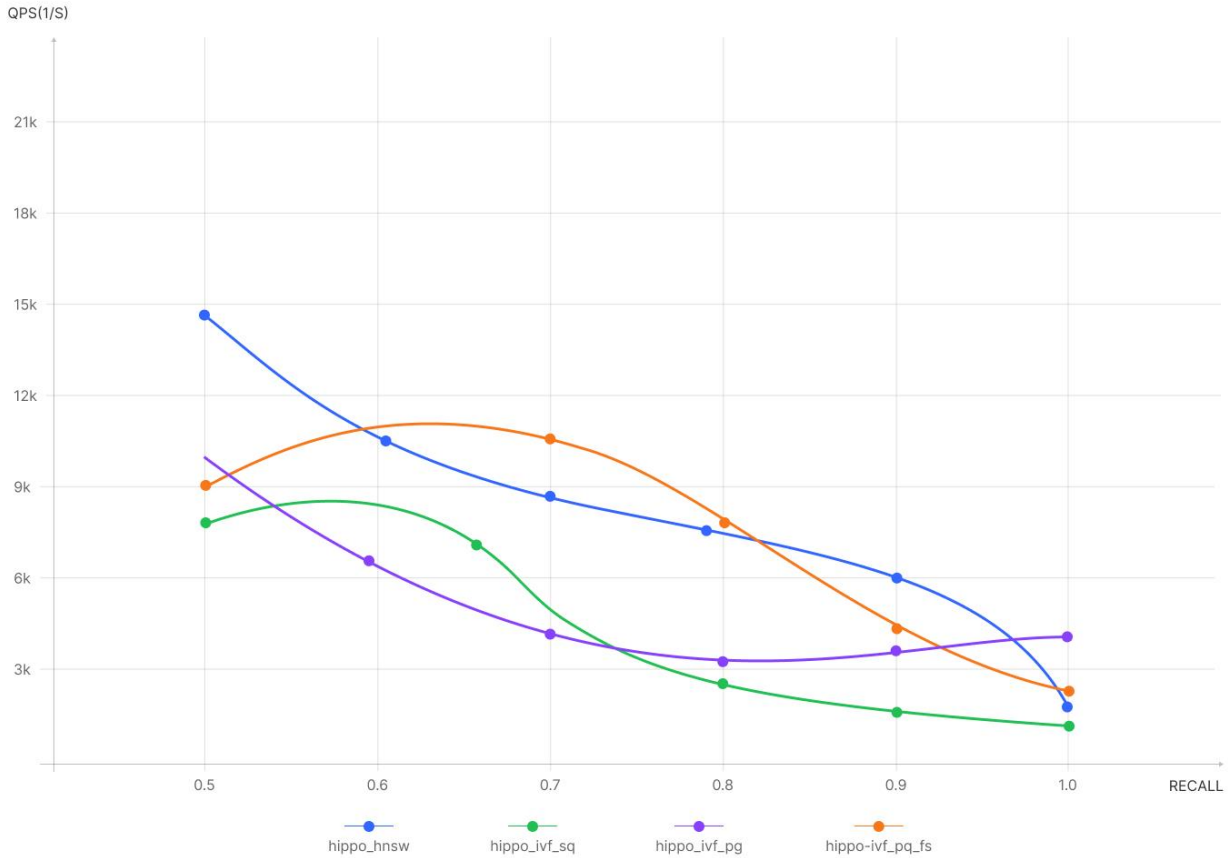
图表 76 Glove-100-angular (k=10) 图

ALGORITHM	PARAMETERS	QPS	95th RT	RECALL
hippo_ivf_pq	Hippo-IVFPQ-refine({'nlist': 4096, 'M': 25}, nprobe: 100, kfactor: 10)	9000	30ms	0.91077
	Hippo-IVFPQ-refine({'nlist': 4096, 'M': 25}, nprobe: 200, kfactor: 100)	8000	31ms	0.95688
	Hippo-IVFPQ-refine({'nlist': 512, 'M': 50}, nprobe: 200, kfactor: 100)	4000	62ms	0.99635
hippo_ivf_pq_fs	Hippo-IVFPQFS-refine({'nlist': 2048, 'M': 50}, nprobe: 200, kfactor: 10)	9000	29ms	0.91696
	Hippo-IVFPQFS-refine({'nlist': 512, 'M': 50}, nprobe: 200, kfactor: 100)	7800	33ms	0.99544
	Hippo-IVFPQFS-refine({'nlist': 512, 'M': 50}, nprobe: 200, kfactor: 1000)	6000	42ms	0.99635
hippo_ivf_sq	Hippo-IVFPQ-refine({'nlist': 4096, 'sq_type': 'SQfp16'}, nprobe: 100, kfactor: 10)	7000	37ms	0.92066
	Hippo-IVFPQ-refine({'nlist': 4096, 'sq_type': 'SQfp16'}, nprobe: 200, kfactor: 10)	5500	44ms	0.95689
	Hippo-IVFPQ-refine({'nlist': 2048, 'sq_type': 'SQfp16'}, nprobe: 200, kfactor: 10)	3900	59ms	0.97377
	Hippo-IVFPQ-refine({'nlist': 512, 'sq_type': 'SQfp16'}, nprobe: 200, kfactor: 1000)	1300	170ms	0.99635

hippo_hnsw	Hippo-HNSW({'M': 36, 'efConstruction': 500}, ef_search: 100)	19500	12ms	0.91299
	Hippo-HNSW({'M': 36, 'efConstruction': 500}, ef_search: 200)	19000	13ms	0.95644
	Hippo-HNSW({'M': 64, 'efConstruction': 500}, ef_search: 800)	6500	41ms	0.99772

图表 77 Glove-100-angular (k=10) 表

7.4.4. Glove-200-angular (k=10)



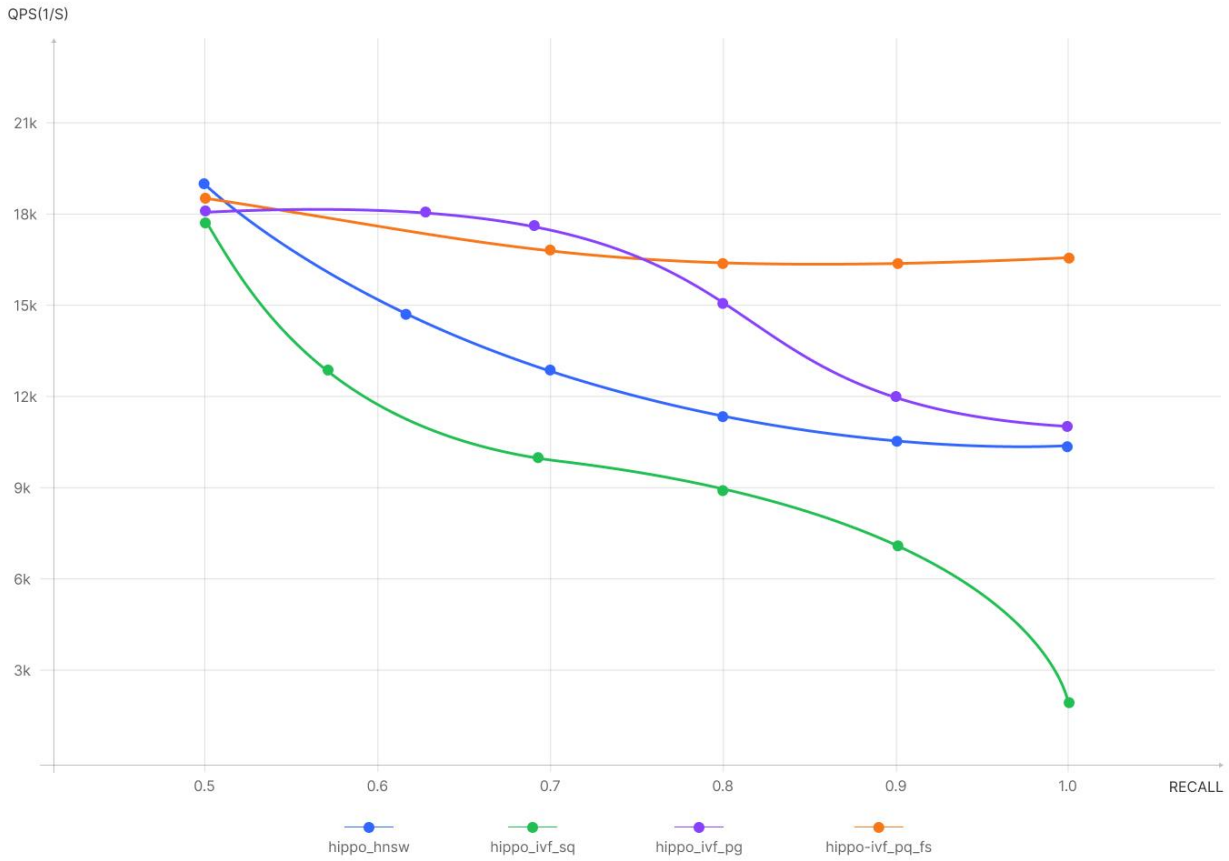
图表 78 Glove-200-angular (k=10) 图

ALGORITHM	PARAMETERS	QPS	95th RT	RECALL
hippo_hnsw	Hippo-HNSW({'M': 64, 'efConstruction': 200}, ef_search: 200)	11500	27ms	0.90876
	Hippo-HNSW({'M': 64, 'efConstruction': 200}, ef_search: 500)	5700	57ms	0.95397
	Hippo-HNSW({'M': 64, 'efConstruction': 500}, ef_search: 800)	2600	120ms	0.98629
hippo_ivf_sq	Hippo-IVFPQ-refine({'nlist': 4096, 'sq_type': 'SQfp16'}, nprobe: 100, kfactor: 10)	6300	45ms	0.8931
	Hippo-IVFPQ-refine({'nlist': 2048, 'sq_type': 'SQfp16'}, nprobe: 200, kfactor: 100)	3300	100ms	0.95672
	Hippo-IVFPQ-refine({'nlist': 512, 'sq_type': 'SQfp16'}, nprobe: 200, kfactor: 10)	1300	260ms	0.99053
	Hippo-IVFPQ-refine({'nlist': 512, 'sq_type': 'SQfp16'}, nprobe: 200, kfactor: 100)	1300	260ms	0.99053

hippo_ivf_pq	Hippo-IVFPQ-refine({'nlist': 4096, 'M': 50}, nprobe: 100, kfactor: 100)	7300	36ms	0.8931
	Hippo-IVFPQ-refine({'nlist': 2048, 'M': 50}, nprobe: 200, kfactor: 100)	7000	37ms	0.9567
	Hippo-IVFPQ-refine({'nlist': 512, 'M': 50}, nprobe: 200, kfactor: 1000)	3800	76ms	0.99053
	Hippo-IVFPQ-refine({'nlist': 512, 'M': 100}, nprobe: 200, kfactor: 1000)	2000	130ms	0.99053
	Hippo-IVFPQ-refine({'nlist': 4096, 'M': 100}, nprobe: 200, kfactor: 1000)	4300	56ms	0.93532
hippo_ivf_pq_fs	Hippo-IVFPQFS-refine({'nlist': 4096, 'M': 100}, nprobe: 200, kfactor: 10)	7800	34ms	0.91573
	Hippo-IVFPQFS-refine({'nlist': 1024, 'M': 100}, nprobe: 200, kfactor: 100)	6600	40ms	0.97697
	Hippo-IVFPQFS-refine({'nlist': 512, 'M': 100}, nprobe: 200, kfactor: 1000)	3800	78ms	0.99053

图表 79 Glove-200-angular (k=10) 表

7.4.5. Sift-128-euclidean (k=10)



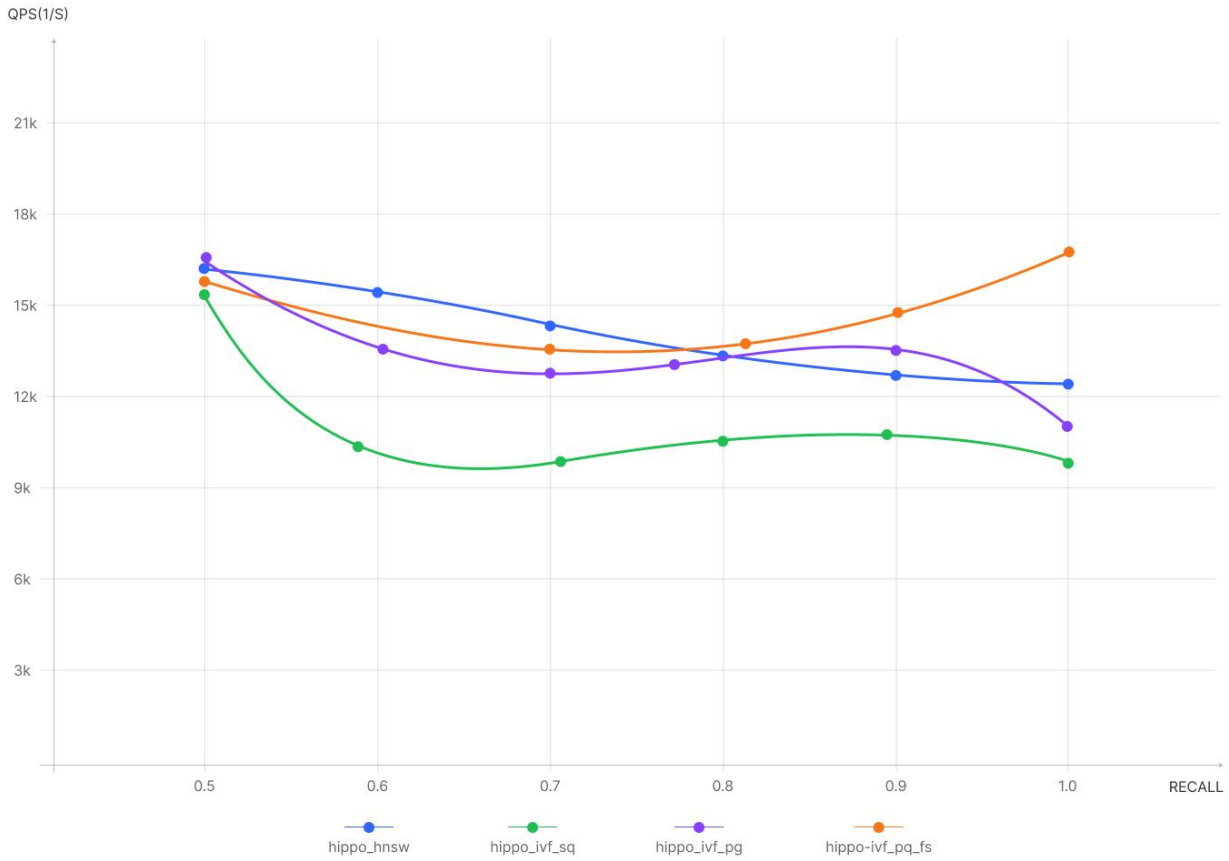
图表 80 Sift-128-euclidean (k=10) 图

ALGORITHM	PARAMETERS	QPS	95th RT	RECALL
hippo_ivf_sq	Hippo-IVFPQ-refine({'nlist': 512, 'sq_type': 'SQfp16'}, nprobe: 10, kfactor: 10)	19000	13ms	0.92045
	Hippo-IVFPQ-refine({'nlist': 2048, 'sq_type': 'SQfp16'}, nprobe: 200, kfactor: 100)	8300	38ms	0.99964
	Hippo-IVFPQ-refine({'nlist': 512, 'sq_type': 'SQfp16'}, nprobe: 200, kfactor: 10)	2600	120ms	1
hippo_ivf_pq	Hippo-IVFPQ-refine({'nlist': 512, 'M': 16}, nprobe: 10, kfactor: 10)	20000	12ms	0.90848

	Hippo-IVFPQ-refine({'nlist': 2048, 'M': 4}, nprobe: 100, kfactor: 100)	19000	13ms	0.95787
	Hippo-IVFPQ-refine({'nlist': 1024, 'M': 32}, nprobe: 200, kfactor: 100)	11000	25ms	0.99998
hippo_ivf_pq_fs	Hippo-IVFPQFS-refine({'nlist': 4096, 'M': 64}, nprobe: 100, kfactor: 10)	20000	12ms	0.97854
	Hippo-IVFPQFS-refine({'nlist': 2048, 'M': 64}, nprobe: 200, kfactor: 100)	19000	13ms	0.99964
	Hippo-IVFPQFS-refine({'nlist': 512, 'M': 64}, nprobe: 200, kfactor: 100)	17000	16ms	1
hippo_hnsw	Hippo-HNSW({'M': 8, 'efConstruction': 500}, ef_search: 50)	21000	11ms	0.9016
	Hippo-HNSW({'M': 48, 'efConstruction': 200}, ef_search: 200)	20000	11ms	0.99911
	Hippo-HNSW({'M': 64, 'efConstruction': 500}, ef_search: 500)	16000	16ms	1

图表 81 Sift-128-euclidean (k=10) 表

7.4.6. Fashion-mnist-784-euclidean (k=10)



图表 82 Fashion-mnist-784-euclidean (k=10) 图

ALGORITHM	PARAMETERS	QPS	95th RT	RECALL
hippo_hnsw	Hippo-HNSW({'M': 64, 'efConstruction': 500}, ef_search: 10)	15000	16ms	0.95258
	Hippo-HNSW({'M': 64, 'efConstruction': 200}, ef_search: 500)	15000	15ms	0.99988
	Hippo-HNSW({'M': 64, 'efConstruction': 500}, ef_search: 800)	13000	18ms	0.99998
hippo_ivf_pq_fs	Hippo-IVFPQFS-refine({'nlist': 512, 'M': 98}, nprobe: 10, kfactor: 10)	15000	14ms	0.9769
	Hippo-IVFPQFS-refine({'nlist': 512, 'M': 98}, nprobe: 100, kfactor: 10)	15000	15ms	0.98829

	Hippo-IVFPQFS-refine({'nlist': 512, 'M': 392}, nprobe: 100, kfactor: 10)	15000	15ms	1
hippo_ivf_pq	Hippo-IVFPQ-refine({'nlist': 1024, 'M': 7}, nprobe: 10, kfactor: 10)	15000	14ms	0.94777
	Hippo-IVFPQ-refine({'nlist': 1024, 'M': 98}, nprobe: 10, kfactor: 10)	15000	15ms	0.96921
	Hippo-IVFPQ-refine({'nlist': 512, 'M': 98}, nprobe: 100, kfactor: 10)	14500	16ms	1
hippo_ivf_sq	Hippo-IVFPQ-refine({'nlist': 2048, 'sq_type': 'SQfp16'}, nprobe: 10, kfactor: 10)	15000	14ms	0.94497
	Hippo-IVFPQ-refine({'nlist': 2048, 'sq_type': 'SQfp16'}, nprobe: 100, kfactor: 10)	15000	15ms	0.99984
	Hippo-IVFPQ-refine({'nlist': 512, 'sq_type': 'SQfp16'}, nprobe: 100, kfactor: 10)	12000	23ms	1

图表 83 Fashion-mnist-784-euclidean (k=10) 表

客户服务

技术支持

感谢您使用星环信息科技（上海）股份有限公司的产品和服务。如您在产品使用或服务中有任何技术问题，可以通过以下途径找到我们的技术人员给予解答。

Email：support@transwarp.io
技术支持热线电话：4007-676-098
官方网址：http://www.transwarp.cn/
论坛支持：http://support.transwarp.cn/

意见反馈

如果您在系统安装，配置和使用中发现任何产品问题，可以通过以下方式反馈：
Email：support@transwarp.io

感谢您的支持和反馈，我们一直在努力！