 下载

<https://github.com/hugegraph>

#cd /home/spark/graph

#tar -zxvf hugegraph-0.10.4.tar.gz

使用scylla作为后端存储

安装好scylla后（在scylla安装文档里面）

#cd /home/spark/graph/hugegraph-0.10.4/conf

#vi hugegraph.properties

backend=scylladb

serializer=scylladb

#cd ../

#bin/init-store.sh

让子弹飞一会

#bin/start-hugegraph.sh

/home/spark/graph/hugegraph-0.10.4/bin/util.sh:行73: lsof: 未找到命令

/home/spark/graph/hugegraph-0.10.4/bin/util.sh:行73: lsof: 未找到命令

Starting HugeGraphServer...

Connecting to HugeGraphServer (http://127.0.0.1:8080/graphs).......OK

让子弹飞一会

验证是否安装成功

#jps

44832 Jps

42581 HugeGraphServer

50581 scylla-jmx-1.0.jar

#echo `curl -o /dev/null -s -w %{http\_code} "http://localhost:8080/graphs/hugegraph/graph/vertices"`

200

远程访问

#cd ../conf

vi rest-server.properties

restserver.url=http://0.0.0.0:8080

重启

#cd ../

#bin/stop-hugegraph.sh

#bin/start-hugegraph.sh

Api介绍：

**K-out API**

功能介绍

根据起始顶点、方向、边的类型（可选）和深度depth，查找从起始顶点出发恰好depth步可达的顶点

Params

source: 起始顶点id，必填项

direction: 起始顶点向外发散的方向（OUT,IN,BOTH），选填项，默认是BOTH

max\_depth: 步数，必填项

label: 边的类型，选填项，默认代表所有edge label

nearest: nearest为true时，代表起始顶点到达结果顶点的最短路径长度为depth，不存在更短的路径；nearest为false时，代表起始顶点到结果顶点有一条长度为depth的路径（未必最短且可以有环），选填项，默认为true

max\_degree: 查询过程中，单个顶点最大边数目，选填项，默认为10000

capacity: 遍历过程中最大的访问的顶点数目，选填项，默认为10000000

limit: 返回的顶点的最大数目，选填项，默认为10000000

链接地址

[http://10.10.10.171:8080/graphs/hugegraph/traversers/kout?source="1:marko"&max\_depth=2](http://10.10.10.171:8080/graphs/hugegraph/traversers/kout?source=%221:marko%22&max_depth=2)

查找恰好N步关系可达的顶点。两个例子：

家族关系中，查找一个人的所有孙子，person A通过连续的两条“儿子”边到达的顶点集合。

社交关系中发现潜在好友，例如：与目标用户相隔两层朋友关系的用户，可以通过连续两条“朋友”边到达的顶点。

**K-neighbor**

功能介绍

根据起始顶点、方向、边的类型（可选）和深度depth，查找包括起始顶点在内、depth步之内可达的所有顶点

相当于：起始顶点、K-out(1)、K-out(2)、… 、K-out(max\_depth)的并集

Params

source: 起始顶点id，必填项

direction: 起始顶点向外发散的方向（OUT,IN,BOTH），选填项，默认是BOTH

max\_depth: 步数，必填项

label: 边的类型，选填项，默认代表所有edge label

max\_degree: 查询过程中，单个顶点最大边数目，选填项，默认为10000

capacity: 遍历过程中最大的访问的顶点数目，选填项，默认为10000000

limit: 返回的顶点的最大数目，选填项，默认为10000000

注意：K-neighbor无需capacity参数，所有访问的顶点都是目标顶点，所以limit就是capacity

[http://10.10.10.171:8080/graphs/hugegraph/traversers/kneighbor?source="1:marko"&max\_depth=2](http://10.10.10.171:8080/graphs/hugegraph/traversers/kneighbor?source=%221:marko%22&max_depth=2)

查找N步以内可达的所有顶点，例如：

家族关系中，查找一个人五服以内所有子孙，person A通过连续的5条“亲子”边到达的顶点集合。

社交关系中发现好友圈子，例如目标用户通过1条、2条、3条“朋友”边可到达的用户可以组成目标用户的朋友圈子

**Shortest Path**

功能介绍

根据起始顶点、目的顶点、方向、边的类型（可选）和最大深度，查找一条最短路径

Params

source: 起始顶点id，必填项

target: 目的顶点id，必填项

direction: 起始顶点向外发散的方向（OUT,IN,BOTH），选填项，默认是BOTH

max\_depth: z最大步数，必填项

label: 边的类型，选填项，默认代表所有edge label

max\_degree: 查询过程中，单个顶点最大边数目，选填项，默认为10000

capacity: 遍历过程中最大的访问的顶点数目，选填项，默认为10000000

[http://10.10.10.171:8080/graphs/hugegraph/traversers/shortestpath?source="1:marko"&target=%"2:ripple"&max\_depth=3](http://10.10.10.171:8080/graphs/hugegraph/traversers/shortestpath?source=%221:marko%22&target=%25%222:ripple%22&max_depth=3)

查找两个顶点间的最短路径，例如：

社交关系网中，查找两个用户有关系的最短路径，即最近的朋友关系链

设备关联网络中，查找两个设备最短的关联关系

**Paths**

功能介绍

根据起始顶点、目的顶点、方向、边的类型（可选）和最大深度等条件查找所有路径

Params

source: 起始顶点id，必填项

target: 目的顶点id，必填项

direction: 起始顶点向外发散的方向（OUT,IN,BOTH），选填项，默认是BOTH

label: 边的类型，选填项，默认代表所有edge label

max\_depth: 步数，必填项

max\_degree: 查询过程中，单个顶点最大边数目，选填项，默认为10000

capacity: 遍历过程中最大的访问的顶点数目，选填项，默认为10000000

limit: 返回的路径的最大数目，选填项，默认为10

[http://10.10.10.171:8080/graphs/hugegraph/traversers/paths?source="1:marko"&target="1:josh"&max\_depth=2](http://10.10.10.171:8080/graphs/hugegraph/traversers/paths?source=%221:marko%22&target=%221:josh%22&max_depth=2)

查找两个顶点间的所有路径，例如：

社交网络中，查找两个用户所有可能的关系路径

设备关联网络中，查找两个设备之间所有的关联路径

**Crosspoints**

功能介绍

根据起始顶点、目的顶点、方向、边的类型（可选）和最大深度等条件查找相交点

Params

source: 起始顶点id，必填项

target: 目的顶点id，必填项

direction: 起始顶点到目的顶点的方向, 目的点到起始点是反方向，BOTH时不考虑方向（OUT,IN,BOTH），选填项，默认是BOTH

label: 边的类型，选填项，默认代表所有edge label

max\_depth: 步数，必填项

max\_degree: 查询过程中，单个顶点最大边数目，选填项，默认为10000

capacity: 遍历过程中最大的访问的顶点数目，选填项，默认为10000000

limit: 返回的交点的最大数目，选填项，默认为10

[http://10.10.10.171:8080/graphs/hugegraph/traversers/crosspoints?source="2:lop"&target="2:ripple"&max\_depth=5&direction=IN](http://10.10.10.171:8080/graphs/hugegraph/traversers/crosspoints?source=%222:lop%22&target=%222:ripple%22&max_depth=5&direction=IN)

查找两个顶点的交点及其路径，例如：

社交网络中，查找两个用户共同关注的话题或者大V

家族关系中，查找共同的祖先

**Rings**

功能介绍

根据起始顶点、方向、边的类型（可选）和最大深度等条件查找可达的环路

例如: 1 -> 25 -> 775 -> 14690 -> 25, 其中环路为 25 -> 775 -> 14690 -> 25

Params

source: 起始顶点id，必填项

direction: 起始顶点发出的边的方向（OUT,IN,BOTH），选填项，默认是BOTH

label: 边的类型，选填项，默认代表所有edge label

max\_depth: 步数，必填项

max\_degree: 查询过程中，单个顶点最大边数目，选填项，默认为10000

capacity: 遍历过程中最大的访问的顶点数目，选填项，默认为10000000

limit: 返回的可达环路的最大数目，选填项，默认为10

<http://10.10.10.171:8080/graphs/hugegraph/traversers/rings?source=%222:lop%22&max_depth=3>

查询起始顶点可达的环路，例如：

风控项目中，查询一个用户可达的循环担保的人或者设备

设备关联网络中，发现一个设备周围的循环引用的设备

**Rays**

功能介绍

根据起始顶点、方向、边的类型（可选）和最大深度等条件查找发散到边界顶点的路径

例如: 1 -> 25 -> 775 -> 14690 -> 2289 -> 18379, 其中 18379 为边界顶点，即没有从 18379 发出的边

Params

source: 起始顶点id，必填项

direction: 起始顶点发出的边的方向（OUT,IN,BOTH），选填项，默认是BOTH

label: 边的类型，选填项，默认代表所有edge label

max\_depth: 步数，必填项

max\_degree: 查询过程中，单个顶点最大边数目，选填项，默认为10000

capacity: 遍历过程中最大的访问的顶点数目，选填项，默认为10000000

limit: 返回的非环路的最大数目，选填项，默认为10

<http://10.10.10.171:8080/graphs/hugegraph/traversers/rays?source=%222:lop%22&max_depth=1&direction=IN>

查找起始顶点到某种关系的边界顶点的路径，例如：

家族关系中，查找一个人到所有还没有孩子的子孙的路径

设备关联网络中，找到某个设备到终端设备的路径

**Customized Paths**

功能介绍

根据一批起始顶点、边规则（包括方向、边的类型和属性过滤）和最大深度等条件查找符合条件的所有的路径

Params

sources: 定义起始顶点，必填项，指定方式包括：

ids：通过顶点id列表提供起始顶点

labels和properties：如果没有指定ids，则使用label和properties的联合条件查询起始顶点

labels：顶点的类型列表

properties：通过属性的值查询起始顶点

注意：properties中的属性值可以是列表，表示只要key对应的value在列表中就可以

steps: 表示从起始顶点走过的路径规则，是一组Step的列表。必填项。每个Step的结构如下：

direction：表示边的方向（OUT,IN,BOTH），默认是BOTH

labels：边的类型列表

properties：通过属性的值过滤边

weight\_by：根据指定的属性计算边的权重，sort\_by不为NONE时有效，与default\_weight互斥

default\_weight：当边没有属性作为权重计算值时，采取的默认权重，sort\_by不为NONE时有效，与weight\_by互斥

degree：查询过程中，单个顶点最大边数目，默认为10000

sample：当需要对某个step的符合条件的边进行采样时设置，-1表示不采样，默认为采样100

sort\_by: 根据路径的权重排序：

NONE表示不排序，默认值

INCR表示按照路径权重的升序排序

DECR表示按照路径权重的降序排序

capacity: 遍历过程中最大的访问的顶点数目，选填项，默认为10000000

limit: 返回的路径的最大数目，选填项，默认为10

with\_vertex：true表示返回结果包含完整的顶点信息（路径中的全部顶点），false时表示只返回顶点id

用postman模拟

post

<http://10.10.10.171:8080/graphs/hugegraph/traversers/customizedpaths>

request body json

{

"sources":{

"ids":[

],

"label":"person",

"properties":{

"name":"marko"

}

},

"steps":[

{

"direction":"OUT",

"labels":[

"knows"

],

"weight\_by":"weight",

"degree":-1

},

{

"direction":"OUT",

"labels":[

"created"

],

"default\_weight":8,

"degree":-1,

"sample":1

}

],

"sort\_by":"INCR",

"with\_vertex":true,

"capacity":-1,

"limit":-1

}

Response body

{

    "paths": [

        {

            "objects": [

                "1:marko",

                "1:josh",

                "2:lop"

            ],

            "weights": [

                1.0,

                8.0

            ]

        }

    ],

    "vertices": [

        {

            "id": "1:marko",

            "label": "person",

            "type": "vertex",

            "properties": {

                "name": "marko",

                "age": 29,

                "city": "Beijing"

            }

        },

        {

            "id": "1:josh",

            "label": "person",

            "type": "vertex",

            "properties": {

                "name": "josh",

                "age": 32,

                "city": "Beijing"

            }

        },

        {

            "id": "2:lop",

            "label": "software",

            "type": "vertex",

            "properties": {

                "name": "lop",

                "lang": "java",

                "price": 328

            }

        }

    ]

}

适合查找各种复杂的路径集合，例如：

社交网络中，查找多个大V的粉丝中看过张艺谋导演的电影的路径（大V—>粉丝---->电影—>张艺谋）

风控网络中，查找多个高风险用户的直系亲属的朋友的路径（高风险用户—>直系亲属—>朋友）

**Customized Crosspoints**

功能介绍

根据一批起始顶点、多种边规则（包括方向、边的类型和属性过滤）和最大深度等条件查找符合条件的所有的路径终点的交集

Params

sources: 定义起始顶点，必填项，指定方式包括：

ids：通过顶点id列表提供起始顶点

labels和properties：如果没有指定ids，则使用label和properties的联合条件查询起始顶点

labels：顶点的类型列表

properties：通过属性的值查询起始顶点

注意：properties中的属性值可以是列表，表示只要key对应的value在列表中就可以

path\_patterns: 表示从起始顶点走过的路径规则，是一组规则的列表。必填项。每个规则是一个PathPattern

每个PathPattern是一组Step列表，每个Step结构如下：

direction：表示边的方向（OUT,IN,BOTH），默认是BOTH

labels：边的类型列表

properties：通过属性的值过滤边

degree：查询过程中，单个顶点最大边数目，默认为10000

capacity: 遍历过程中最大的访问的顶点数目，选填项，默认为10000000

limit: 返回的路径的最大数目，选填项，默认为10

with\_path：true表示返回交点所在的路径，false表示不返回交点所在的路径

with\_vertex：

true表示返回结果包含完整的顶点信息（路径中的全部顶点）

with\_path为true时，返回所有路径中的顶点的完整信息

with\_path为false时，返回所有交点的完整信息

false时表示只返回顶点id

用postman模拟

post

http://10.10.10.171:8080/graphs/hugegraph/traversers/customizedcrosspoints

request body json

response body

{

"sources":{

"ids":[

"2:lop",

"2:ripple"

]

},

"path\_patterns":[

{

"steps":[

{

"direction":"IN",

"labels":[

"created"

],

"degree":-1

}

]

}

],

"with\_path":true,

"with\_vertex":true,

"capacity":-1,

"limit":-1

}

{

    "crosspoints": [

        "1:josh"

    ],

    "paths": [

        {

            "objects": [

                "2:ripple",

                "1:josh"

            ]

        },

        {

            "objects": [

                "2:lop",

                "1:josh"

            ]

        }

    ],

    "vertices": [

        {

            "id": "2:ripple",

            "label": "software",

            "type": "vertex",

            "properties": {

                "name": "ripple",

                "lang": "java",

                "price": 199

            }

        },

        {

            "id": "1:josh",

            "label": "person",

            "type": "vertex",

            "properties": {

                "name": "josh",

                "age": 32,

                "city": "Beijing"

            }

        },

        {

            "id": "2:lop",

            "label": "software",

            "type": "vertex",

            "properties": {

                "name": "lop",

                "lang": "java",

                "price": 328

            }

        }

    ]

}

查询一组顶点通过多种路径在终点有交集的情况。例如：

在商品图谱中，多款手机、学习机、游戏机通过不同的低级别的类目路径，最终都属于一级类目的电子设备

**Vertices**

2.10.1 根据顶点的id列表，批量查询顶点

Params

ids: 要查询的顶点id列表

<http://10.10.10.171:8080/graphs/hugegraph/traversers/vertices?ids=%221:marko%22&ids=%222:lop%22>

**获取顶点 Shard 信息**

通过指定的分片大小split\_size，获取顶点分片信息（可以与 3.10.3 中的 Scan 配合使用来获取顶点）。

Params

split\_size: 分片大小

<http://10.10.10.171:8080/graphs/hugegraph/traversers/vertices/scan?start=0&end=4294967295>

按id列表查询顶点比较简单，不再赘述

获取分片和按分片查询顶点，可以用来遍历全部顶点

**Edges**

根据边的id列表，批量查询边

Params

ids: 要查询的边id列表

<http://10.10.10.171:8080/graphs/hugegraph/traversers/edges?ids=%22S1:josh%3E1%3E%3ES2:lop%22&ids=%22S1:josh%3E1%3E%3ES2:ripple%22>

#### 获取边 Shard 信息

通过指定的分片大小split\_size，获取边分片信息（可以与 3.11.3 中的 Scan 配合使用来获取边）。

Params

split\_size: 分片大小

<http://10.10.10.171:8080/graphs/hugegraph/traversers/edges/shards?split_size=4294967295>

#### 根据 Shard 信息批量获取边

通过指定的分片信息批量查询边（Shard信息的获取参见 3.11.2）。

Params

start: 分片起始位置

end: 分片结束位置

page：分页位置，默认为null，不分页；当page为“”时表示分页的第一页，从start指示的位置开始

<http://10.10.10.171:8080/graphs/hugegraph/traversers/edges/scan?start=0&end=3221225469>

按id列表查询边比较简单，不再赘述

获取分片和按分片查询边，可以用来遍历全部边