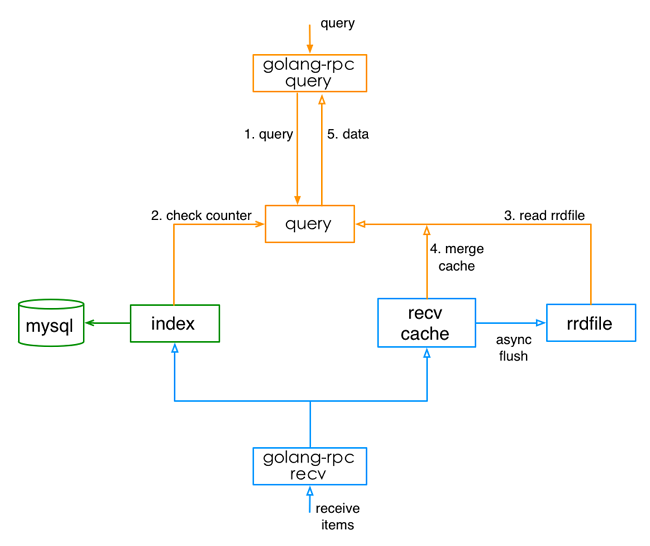
## graph

### debug/api

http://192.168.0.179:6060/debug/connpool/graph

### graph主要的模块结构（含数据流）



### 点击进入图表页面

E:\workspace\open-falcon\dashboard\rrd\static\js\xperf.js

$.ajax({  
 url: **"/chart"**,  
 dataType: **"json"**,  
 method: **"POST"**,  
 data: {**"endpoints"**: checked\_hosts, **"counters"**: checked\_items, **"graph\_type"**: **"h"**, **"\_r"**: Math.random()},  
 success: **function**(ret) {  
 **if** (ret.ok) {  
 setTimeout(**function**(){w.location=**'/chart/big?id='**+ret.id;}, 0);  
 } **else** {  
 alert(**"请求出错了"**);  
 }  
 },  
 error: **function**(){  
 alert(**"请求出错了"**);  
 }  
});

<http://192.168.1.55:8081/chart/h?-=0.34642552165314555&cf=AVERAGE&graph_type=h&id=365244&start=-300&sum=off&sumonly=off&tongbi>=

E:\workspace\open-falcon\dashboard\rrd\view\chart.py

query\_result = graph\_query(endpoint\_counters, g.cf, g.start, g.end)

E:\workspace\open-falcon\dashboard\rrd\utils\rrdgraph.py

r = requests.post(**"%s/graph/history"** %QUERY\_ADDR, data=json.dumps(params))

### 重建索引

2.进行一次索引数据的全量更新。方法为 curl -s "$Hostname.Of.Task:$Http.Port/index/updateAll"。这里，"$Hostname.Of.Task:$Http.Port"是task的http接口地址。 PS:索引数据存放在graph实例上，这里，只是通过task，触发了各个graph实例的索引全量更新。更直接的办法，是，到每个graph实例上，运行curl -s "127.0.0.1:6071/index/updateAll"，直接触发graph实例 进行索引全量更新(这里假设graph的http监听端口为6071)。

### Index/cache.go根据endpoint和counter获取键值

pk := cutils.Md5(fmt.Sprintf("%s/%s", endpoint, counter))

### Graph调试

graph以http的方式提供了多个调试接口。主要有 内部状态统计接口、历史数据查询接口等。脚本./test/debug将一些接口封装成了shell的形式，可自行查阅代码、不在此做介绍。

历史数据查询接口HTTP:GET, curl -s "http://hostname:port/history/$endpoint/$metric/$tags"，返回graph接收到的、最新的3个数据。

# history没有tags的数据,$endpoint=test.host, $metric=agent.alive

curl -s "http://127.0.0.1:6071/history/test.host/agent.alive" | python -m json.tool

# history有tags的数据,$tags='module=graph,pdl=falcon'

curl -s "http://127.0.0.1:6071/history/test.host/qps/module=graph,pdl=falcon" | python -m json.tool

内部状态统计接口HTTP:GET, curl -s "http://hostname:port/statistics/all"，输出json格式的内部状态数据，格式如下。这些内部状态数据，被task组件采集后push到falcon系统，用于绘图展示、报警等。

curl -s "http://127.0.0.1:6071/statistics/all" | python -m json.tool

# output

{

"data": [

{ // counter of received items

"Cnt": 7, // cnt

"Name": "GraphRpcRecvCnt", // name of counter

"Other": {}, // other infos

"Qps": 0, // growth rate of this counter, per second

"Time": "2015-06-18 12:20:06" // time when this counter takes place

},

{ // counter of query requests graph received

"Cnt": 0,

"Name": "GraphQueryCnt",

"Other": {},

"Qps": 0,

"Time": "2015-06-18 12:20:06"

},

{ // counter of all sent items in query

"Cnt": 0,

"Name": "GraphQueryItemCnt",

"Other": {},

"Qps": 0,

"Time": "2015-06-18 12:20:06"

},

{ // counter of info requests graph received

"Cnt": 0,

"Name": "GraphInfoCnt",

"Other": {},

"Qps": 0,

"Time": "2015-06-18 12:20:06"

},

{ // counter of last requests graph received

"Cnt": 3,

"Name": "GraphLastCnt",

"Other": {},

"Qps": 0,

"Time": "2015-06-18 12:20:06"

},

{ // counter of index updates

"Cnt": 0,

"Name": "IndexUpdateAllCnt",

"Other": {},

"Time": "2015-06-18 10:58:52"

}

],

"msg": "success"

}

### 根据endpoint和counter读取文件格式

007ad1a2c9d733a6f8d672fa8a1954bf\_GAUGE\_60.rrd

Api/graph

#Key为endpoint和counter

md5 := cutils.Md5(param.Endpoint + "/" + param.Counter)

    filename := fmt.Sprintf("%s/%s/%s\_%s\_%d.rrd", g.Config().RRD.Storage, md5[0:2], md5, dsType, step)

# Api/graph

#根据endpoint和counter获取dsType和step

dsType, step, exists := index.GetTypeAndStep(param.Endpoint, param.Counter)

### ~~通过api获取两天的AVERAGE数据(无效)~~

curl -s "127.0.0.1:6071/v2/history?e=192.168.1.136&m=cpu.busy&type=AVERAGE&step=172800" | python -m json.tool

curl -s "127.0.0.1:6071/v2/history?e=192.168.1.136&m=cpu.busy&type=AVERAGE&step=3600" | python -m json.tool

实际并不生效

"data": [

{

"dstype": "GAUGE",

"endpoint": "192.168.1.136",

"heartbeat": 120,

"max": "U",

"metric": "cpu.busy",

"min": "U",

"step": 60,

"tags": {},

"timestamp": 1472004480,

"value": 4.5

},

### Api/graph.go

### @graph/history/accurate 网络无数据

18:37:58  
罗岸 2016/9/1 18:37:58  
  
{"cf":"GAUGE","end":1472715124,"endpoint\_counters":[{"counter":"cpu.busy","endpoint":"192.168.0.182"},{"counter":"cpu.system","endpoint":"192.168.0.182"},{"counter":"cpu.user","endpoint":"192.168.0.182"},{"counter":"mem.memtotal","endpoint":"192.168.0.182"},{"counter":"mem.memused","endpoint":"192.168.0.182"},{"counter":"mem.memused.percent","endpoint":"192.168.0.182"},{"counter":"net.if.in.bytes/iface=eno16777984","endpoint":"192.168.0.182"},{"counter":"net.if.out.bytes/iface=eno16777984","endpoint":"192.168.0.182"},{"counter":"disk.io.read\_bytes/device=sda","endpoint":"192.168.0.182"},{"counter":"disk.io.read\_bytes/device=sdb","endpoint":"192.168.0.182"},{"counter":"disk.io.write\_bytes/device=sda","endpoint":"192.168.0.182"},{"counter":"disk.io.write\_bytes/device=sdb","endpoint":"192.168.0.182"}],"start":1472711824,"step":60}

E:\workspace\yh\OpenBridge-passos-proxy\open-faclon\src\github.com\open-falcon\graph\api\graph.go

//用于定位dsType，有两个值GAUGE和DERIVE

    dsType, \_, exists := index.GetTypeAndStep(param.Endpoint, param.Counter) // complete dsType and step

**if** !exists {

**return** nil

    }

//locate the file

    filename := g.RrdFileName(cfg.RRD.Storage, md5, dsType, 60)

#有显示网络数据

http://192.168.0.179:9966/graph/history

{

"cf" : "AVERAGE",

"end" : 1472715124,

"start" : 1472711824,

"step" : 60,

"endpoint\_counters" : [{

"counter" : "cpu.busy",

"endpoint" : "192.168.0.182"

}, {

"counter" : "cpu.system",

"endpoint" : "192.168.0.182"

}, {

"counter" : "cpu.user",

"endpoint" : "192.168.0.182"

}, {

"counter" : "mem.memtotal",

"endpoint" : "192.168.0.182"

}, {

"counter" : "mem.memused",

"endpoint" : "192.168.0.182"

}, {

"counter" : "mem.memused.percent",

"endpoint" : "192.168.0.182"

}, {

"counter" : "net.if.in.bytes/iface=eno16777984",

"endpoint" : "192.168.0.182"

}, {

"counter" : "net.if.out.bytes/iface=eno16777984",

"endpoint" : "192.168.0.182"

}, {

"counter" : "disk.io.read\_bytes/device=sda",

"endpoint" : "192.168.0.182"

}, {

"counter" : "disk.io.read\_bytes/device=sdb",

"endpoint" : "192.168.0.182"

}, {

"counter" : "disk.io.write\_bytes/device=sda",

"endpoint" : "192.168.0.182"

}, {

"counter" : "disk.io.write\_bytes/device=sdb",

"endpoint" : "192.168.0.182"

}

]

}

### @提供step接口graph/history/accurate

测试用例

http://192.168.1.55:9966/graph/history/accurate

{

"start": 1472040000,

"end": 1472083200,

"cf": "AVERAGE",

"step": 43200,

"endpoint\_counters": [

{

"endpoint" : "192.168.1.136",

"counter" : "cpu.busy"

},

{

"endpoint" : "192.168.1.137",

"counter" : "cpu.busy"

}

]

}

Api/graph.go

**func** (this \*Graph) AccurateQuery(param cmodel.GraphAccurateQueryParam, resp \*cmodel.GraphQueryResponse) error {

**var** datas []\*cmodel.RRDData

    // statistics

    proc.GraphQueryCnt.Incr()

    cfg := g.Config()

    // form empty response

    resp.Values = []\*cmodel.RRDData{}

    resp.Endpoint = param.Endpoint

    resp.Counter = param.Counter

    resp.DsType = param.ConsolFun

    resp.Step = param.Step

    start\_ts := param.Start - param.Start%int64(param.Step)

    end\_ts := param.End - param.End%int64(param.Step) + int64(param.Step)

**if** end\_ts-start\_ts-int64(param.Step) < 1 {

**return** nil

    }

    md5 := cutils.Md5(param.Endpoint + "/" + param.Counter)

    //locate the file

    filename := g.RrdFileName(cfg.RRD.Storage, md5, "GAUGE", 60)

    // read data from rrd file

    datas, \_ = rrdtool.Fetch(filename, param.ConsolFun, start\_ts, end\_ts, param.Step)

    resp.Values = datas

    // statistics

    proc.GraphQueryItemCnt.IncrBy(int64(len(resp.Values)))

**return** nil

}

Common\model\graph.go

// 页面上已经可以看到DsType和Step了，直接带进查询条件，Graph更易处理

**type** GraphAccurateQueryParam **struct** {

    Checksum string `json:"checksum"`

    Start int64 `json:"start"`

    End int64 `json:"end"`

    ConsolFun string `json:"consolFuc"`

    Endpoint string `json:"endpoint"`

    Counter string `json:"counter"`

    DsType string `json:"dsType"`

    Step int `json:"step"`

}

E:\workspace\open-falcon\test\src\graph\api\graph.go

E:\workspace\open-falcon\test\src\github.com\open-falcon\common\model\graph.go

E:\workspace\open-falcon\test\src\query\graph\graph.go

E:\workspace\open-falcon\test\src\query\http\graph\_http.go

### AccurateQuery

E:\workspace\yh\OpenBridge-passos-proxy\open-faclon\src\graph\api\graph.go

# 增加step参数，可用于控制查询的精度 by qlyzpqz · Pull Request #33 · open-falcon/graph

https://github.com/open-falcon/graph/pull/33

1、增加step参数，用于query控制查询的精度  
2、修复start % step == 0时，第1点数据取值为0的问题  
3、查询数据时，将内存中的数据，也进行归档，将归档的数据补充在最后，返回给query

# fix graph migration 'no such file or directory' rpc\_call error bug by KevinTHU · Pull Request #36 · open-falcon/graph

https://github.com/open-falcon/graph/pull/36

**描述：**  
进行graph迁移时，会出现migrate.go:301: open /opt/open-falcon/graph/data/6070/35/356aa82f9c00c1edd11102696313c636\_GAUGE\_60.rrd: no such file or directory这样的报错，并且该报错会持续发生，每次落盘周期30分钟一次，经查证，为代码段：  
if os.IsNotExist(err) {  
并未成功判断该err为os.IsNotExist（“文件不存在”）这个错误类型所造成，查看go源码，发现os.IsNotExist只针对PathError有效，可判断其是否为os.IsNotExist，但对rpc\_call返回的普通error无法判断，造成if语句无法按预期判断，迁移时，一旦有新的指标项传入新的graph，则会报fetch\_s\_error，并无法落盘，只能写入内存。  
**可能的隐患：**  
如果迁移长时间持续进行，中途未进行重启graph的操作，新的指标监控项会造成内存数据持续增加，直至OOM，并且会造成log日志持续写入，日志增长过快。  
**解决方式：**  
修改if os.IsNotExist(err) {为if strings.Contains(err.Error(), "no such file or directory")，直接对rpc\_call返回的error进行内容判断，经测试验证通过

# Api/graph.go

## Query

### rralStarTs理解

nowTs := time.Now().Unix()//获取当前时间  
lastUpTs := nowTs - nowTs%int64(step)//数据清理  
rra1StartTs := lastUpTs - int64(rrdtool.*RRA1PointCnt*\*step)//确定最后记录时间

如果当前时间已超过查询范围，表明数据已经保存在文件中，不用从内存中查询

// consolidated, do not merge  
**if** start\_ts < rra1StartTs {  
 resp.Values = datas  
 **goto** \_RETURN\_OK  
}