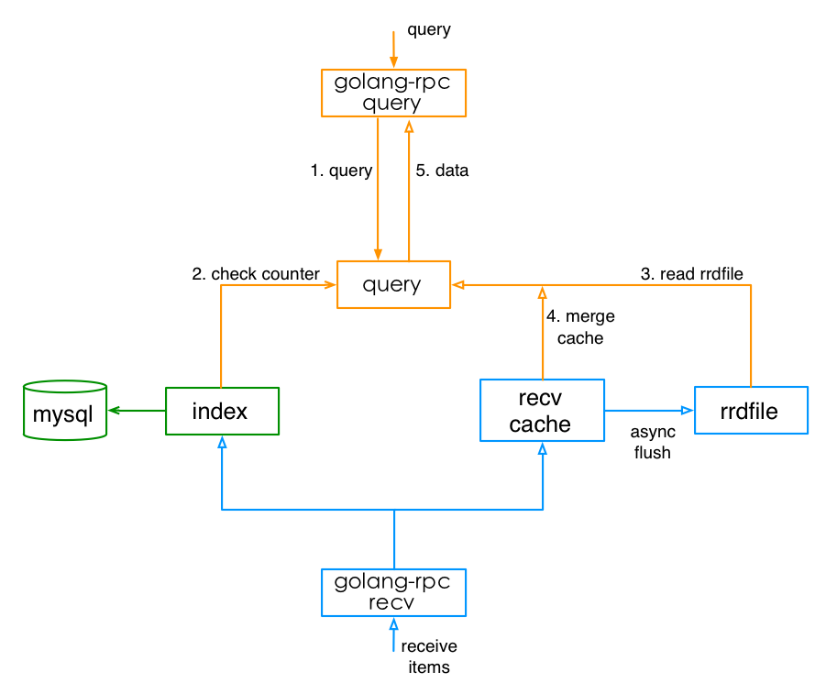
在不引入influxdb时，open-falcon的存储方式：

transfer整理监控项数值，做一致性hash分片，传输给graph以进行数据的存储

graph收到监控项数据后，将数据存储成RRD文件格式，进行归档，同时提供查询RPC接口

api面向终端用户，收到查询请求后，会去多个graph里面，查询不同metric的数据，汇总后将监控数据传送到dashboard以进行页面展示



graph组件，使用rrdtool，来存储监控历史数据。主要功能，如下：

0. 提供Golang-RPC接口，接收格式化的监控数据。监控数据的格式，见[这里](http://blog.niean.name/2015/08/06/falcon-intro/#数据模型)。

1. 支持存储原始数据。默认保存12H的原始数据。

2. 支持对原始数据进行归档存储。支持5min、20min、3h、12h四个归档粒度。

3. 提供Http-Get和Golang-Rpc两种接口，用于查询历史数据。查询数据时，graph会自适应选择归档粒度，使返回的数据点不过多。

#### 数据接收

graph对外提供了golang-rpc形式的接口，用于接收外部push过来的监控数据。收到的数据，被复制到索引生成器`index`一份、被复制到接收缓存`recv cache`一份。索引生成器，会为新增的监控数据建立一条索引信息（形如`endpoint/metric/tags`），并存储到外部的mysql数据库中、供dashboard等系统使用。接收缓存，是监控数据被写入磁盘`rrdfile`前的一个缓存，目的是优化磁盘写入性能。

#### 数据查询

graph对外提供了golang-rpc形式的接口，用于接收外部的查询请求。query模块，会使用索引信息来验证外部请求的counter合法性，验证通过后，读取磁盘上`rrdfile`内容、并merge接收缓存中尚未flush到磁盘的监控数据，得到完整的结果后 返回给外部。

## 扩容

----

系统扩容时，只需求改transfer和query的graph集群配置即可，graph组件会自动完成历史数据迁移的工作。扩容期间，graph可以正常的对外提供服务。

## 缺点

----

1. 对磁盘资源消耗严重。rrdtool自带的归档功能，会消耗大量的磁盘IO。

2. 精确的历史数据保存时间短，不利于历史的现场回放。默认只保存12H的原始数据。

3. 绘图数据的高可用，实现成本较高。冷备、热备绘图数据都多多少少存在一些问题，灾难恢复也可能需要较多的时间。

对于监控系统来讲，历史数据的存储和高效率查询，永远是个很难的问题！

数据量大：目前我们的监控系统，每个周期，大概有 2000 万次数据上报（上报周期为 1

分钟和 5 分钟两种，各占 50%），一天 24 小时里，从来不会有业务低峰，不管是白天和黑夜，每个周期，总会有那么多的数据要更新。

写操作多：一般的业务系统，通常都是读多写少，可以方便的使用各种缓存技术，再者

各类数据库，对于查询操作的处理效率远远高于写操作。而监控系统恰恰相反，写操作远远

高于读。每个周期几千万次的更新操作，对于常用数据库（MySQL、postgresql、mongodb）都是无法完成的。

高效率的查：我们说监控系统读操作少，是说相对写入来讲。监控系统本身对于读的要

求很高，用户经常会有查询上百个 meitric，在过去一天、一周、一月、一年的数据。如何在 1 秒内返回给用户并绘图，这是一个不小的挑战

open-falcon 在这块，投入了较大的精力。我们把数据按照用途分成两类，一类是用来绘

图的，一类是用户做数据挖掘的。

对于绘图的数据来讲，查询要快是关键，同时不能丢失信息量。对于用户要查询 100 个

metric，在过去一年里的数据时，数据量本身就在那里了，很难 1 秒之类能返回，另外就算返\回了，前端也无法渲染这么多的数据，还得采样，造成很多无谓的消耗和浪费。我们参考

rrdtool 的理念，在数据每次存入的时候，会自动进行采样、归档。

我们的归档策略如下，历史数据保存 5 年。同时为了不丢失信息量，数据归档的时候，会按照平均值采样、最大值采样、最小值采样存三份。