

基于Prometheus的数据库监控

原创

Linux操作系统

作者：沃趣科技

时间：2017-04-21 14:06:12

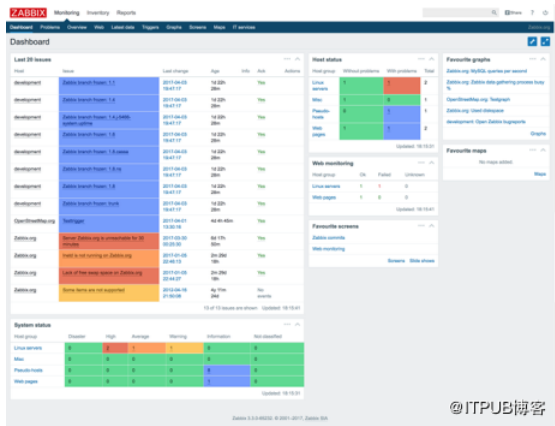
1851

0

基于Prometheus的数据库监控
传统监控系统面临的问题
Prometheus的前身：Borgmon
Borgmon介绍
应用埋点
服务发现
指标采集与堆叠
指标数据存储
指标
指标的查询
规则计算
Prometheus
介绍
架构
数据库监控
部署服务端
部署exporter端

基于Prometheus的数据库监控 传统监控系统面临的问题

传统监控系统，会面临哪些问题？
以zabbix为例



初次使用需要配置大量管理功能，随着服务器和业务的增长会发现zabbix，这种传统监控面临很多问题

- DB性能瓶颈，由于zabbix会将采集到的性能指标都存储到数据库中，当服务器数量和业务增长很快时数据库性能首先成为瓶颈。
- 多套部署，管理成本高，当数据库性能成为瓶颈时首先想到的办法可能时分多套zabbix部署，但是又会带来管理很维护成本很高的问题。
- 易用性差，zabbix的配置和管理非常复杂，很难精通。
- 邮件风暴，邮件配置各种规则相当复杂，一不小心可能就容易造成邮件风暴的问题。

随着容器技术的发展，传统监控系统面临更多问题

- 容器如何监控？
- 微服务如何监控？
- 集群性能如何进行分析计算？
- 如何管理agent端大量配置脚本？

我们可以看到传统监控系统无法满足，当前IT环境下的监控需求

Prometheus的前身：Borgmon

2015年Google发表了一篇文章《Google使用Borg进行大规模集群的管理》

沃趣科技

沃趣科技

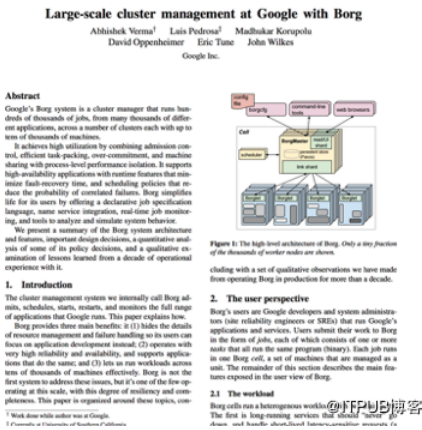
专家姓名：沃趣科技

杭州沃趣科技股份有限公司创建于2012年（股票代码：839849），是一家专注为企业用户提供基于高性能、高可用、可扩展的开放数据库云平台解决方案的国产厂商。公司创始团队为原阿里巴巴数据库技术团队核心骨干，凭借丰富的研发及运维经验，为行业客户提供数据库云产品及软硬件一体化解决方案。

注册时间：2016-07-18

博文量170 | 访问量611151

- 博文推荐
- Linux服务器---关闭selinux
一生有你llx
- Linux基础命令---diff
一生有你llx
- Linux基础命令---rmdir
一生有你llx
- Linux基础命令---chown
一生有你llx
- Linux基础命令---chmod
一生有你llx
- Linux基础命令---e2image
一生有你llx
- Linux基础命令---tune2fs
一生有你llx
- Linux基础命令---dumpe2fs
一生有你llx
- Linux基础命令---swapoff
一生有你llx
- Linux基础命令---e2fsck
一生有你llx



这篇论文也描述了Google集群的规模和面临的挑战

- 单集群上百万服务器
- 几千个不同的应用
- 几十万个以上的jobs，而且动态增加或者减少
- 每个数据中心数百个集群

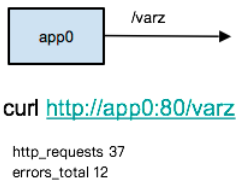
基于这样一个规模，Google的监控系统也面临巨大挑战，而Borg中的Borgmon监控系统就是为了应对这些挑战而生。

Borgmon介绍

那么我们来看一下Google如何做大规模集群的监控系统

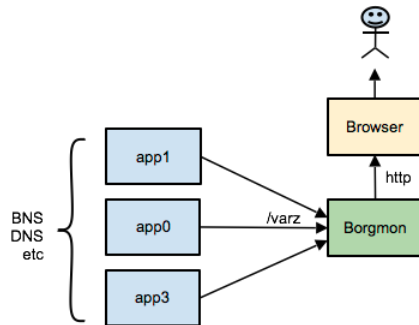
应用埋点

首先，Borg集群中运行的所有应用都需要暴露出特定的URL，http://<app>:80/varz 通过这个URL我们就可以获取到应用所暴露的全部监控指标。



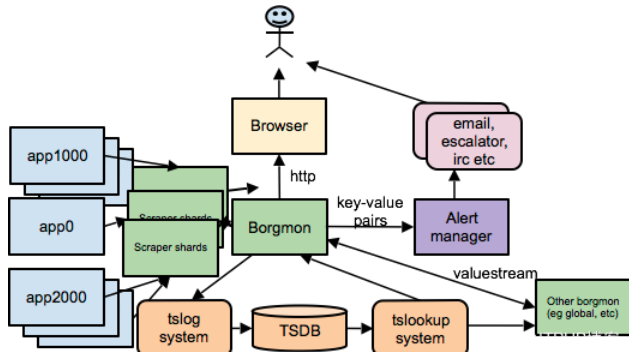
服务发现

然而这样的应用有数千万个，而且可能会动态增加或者减少，Borgmon中如何发现这些应用呢？Borg中的应用启动时会自动注册到Borg内部的域名服务器BNS中，Borgmon通过读取BNS中应用列表信息，收集到应用列表，从而发现有哪些应用服务需要监控。当获取到应用列表后，就会将应用的全部监控变量值拉取到Borgmon系统中。



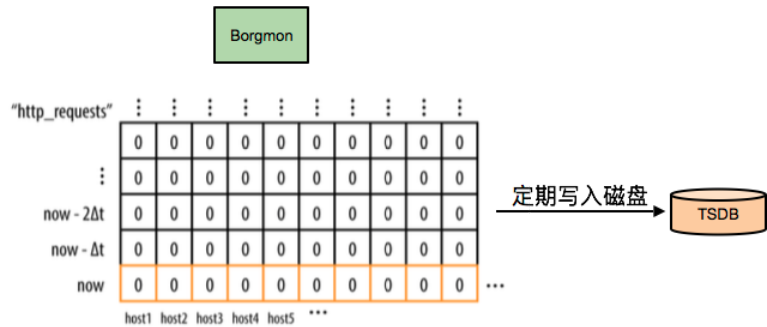
指标采集与堆叠

当监控指标收集到Borgmon中，就可以进行展现或者提供给告警使用，另外由于一个集群实在是太庞大了，一个Borgmon可能无法满足整个集群的监控采集和展现需求，所以一般会在一些复杂的环境下，一个数据中心可能部署多个Borgmon，分为数据收集层和汇总层，数据收集层会有多个Borgmon专门用来到应用中收集数据，汇总层Borgmon则从数据收集层Borgmon中获取数据。



指标数据存储

Borgmon收集到了性能指标数据后，会把所有的数据存储在内存数据库里，定时checkpoint到磁盘上，并且会周期性的打包到外部的系统TSDB。通常情况下，数据中心和全局Borgmon中一般至少会存放12小时左右的数据量，以便渲染图表使用。每个数据点大概占用24字节的内存，所以存放100万个time-series，每个time-series每分钟一个数据点，同时保存12小时数据，仅需17GB内存。



指标

指标的查询

Borgmon中通过标签的方式查询指标，基于标签过滤我们可以查询到某个应用的具体指标，也可以查询更高维度的信息
基于标签过滤信息，比如我们基于一组过滤信息查询到host0:80这个app的http_requests

```
{var=http_requests,job=webserver,instance=host0:80,service=web,zone=us-west}@ITPUB博客
```

我们也可以查询到整个美国西部，job为webserver的http_requests

```
{var=http_requests,job=webserver,service=web,zone=us-west}@ITPUB博客
```

那么这个时候拿到的就是所有符合条件的实例的列表

```
{var=http_requests,job=webserver,instance=host0:80,service=web,zone=us-west} 10
{var=http_requests,job=webserver,instance=host1:80,service=web,zone=us-west} 9
{var=http_requests,job=webserver,instance=host2:80,service=web,zone=us-west} 11
{var=http_requests,job=webserver,instance=host3:80,service=web,zone=us-west} 0
{var=http_requests,job=webserver,instance=host4:80,service=web,zone=us-west} 10 @ITPUB博客
```

规则计算

在数据收集和存储的基础之上，我们可以通过规则计算得到进一步的数据。

比如，我们想在web server报错超过一定比例的时候报警，或者说在非200返回码，占总请求的比例超过某个值的时候报警。

```
rules <<<
  # Compute the rate of requests for each task from the count of requests
  {var=task:http_requests:rate10m,job=webserver} =
    rate({var=http_requests,job=webserver}[10m]);
  # Sum the rates to get the aggregate rate of queries for the cluster;
  # 'without instance' instructs Borgmon to remove the instance label
  # from the right hand side.
  {var=dc:http_requests:rate10m,job=webserver} =
    sum without instance({var=task:http_requests:rate10m,job=webserver})
  >>> @ITPUB博客
```

```
rules <<<
  # Compute the rate of requests for each task from the count of requests
  {var=task:http_requests:rate10m,job=webserver} =
    rate({var=http_requests,job=webserver}[10m]);
  # Sum the rates to get the aggregate rate of queries for the cluster;
  # 'without instance' instructs Borgmon to remove the instance label
  # from the right hand side.
  {var=dc:http_requests:rate10m,job=webserver} =
    sum without instance({var=task:http_requests:rate10m,job=webserver})
  >>> @ITPUB博客
```

Prometheus

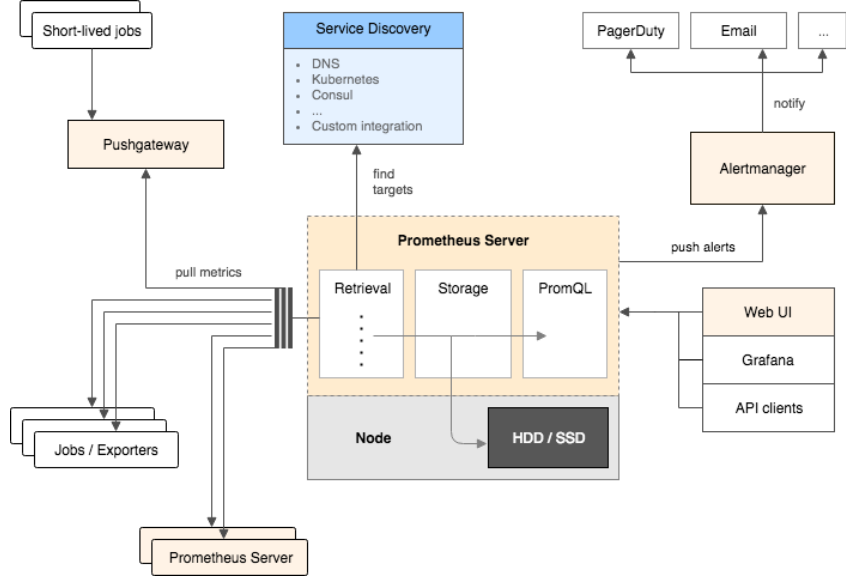
介绍

Borgmon是Google内部的系统，那么在Google之外如何使用它呢？这里就提到我们所描述的Prometheus这套监控系统。
Google内部SRE工程师的著作《Google SRE》这本书中，直接就提到了Prometheus相当于就是开源版本的Borgmon。目前Prometheus在开源社区也是相当火爆，由Google发起Linux基金会旗下的原生云基金会（CNCF）就将Prometheus纳入其下第二大开源项目（第一项目为Kubernetes，为Borg的开源版本）。

架构

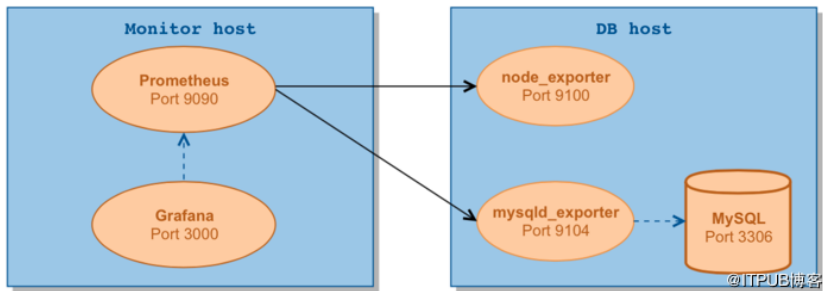
Prometheus整体架构和Borgmon类似，组件如下，有些组件是可选的：

- Prometheus主服务器，用来收集和存储时间序列数据
- 应用程序client代码库
- 短时jobs的push gateway
- 特殊用途的exporter（包括HAProxy、StatsD、Ganglia等）
- 用于报警的alertmanager
- 命令行工具查询
- 另外Grafana是作为Prometheus Dashboard展现的绝佳工具



数据库监控

基于Prometheus的数据库指标采集，我们以MySQL为例，由于MySQL没有暴露采集性能指标的接口，我们可以单独启动一个mysql_exporter，通过mysql_exporter到MySQL数据库上抓去性能指标，并暴露出性能采集接口提供给Prometheus，另外我们可以启动node_exporter用于抓取主机的性能指标。



部署服务端

对于服务端配置非常简单，由于Prometheus全部基于Go语言开发，而Go语言程序在安装方面非常方便，安装服务端程序只需要下载，解压并运行即可。可以看到服务端常用程序也比较少，只需要包含prometheus这个主服务程序和alertmanager这个告警系统程序。

```
[root@server prometheus]# tree -L 1
.
├── alertmanager
├── alertmanager.yml
├── host.yml
├── LICENSE
├── mysql.yml
├── NOTICE
├── prometheus
└── prometheus.yml

0 directories, 8 files
```

服务端配置也非常简单，常用配置包含拉取时间和具体采集方式，就我们监控mysql数据库来讲，只需要填入mysql_exporter地址即可。

```
[root@server prometheus]# cat prometheus.yml
global:
  evaluation_interval: 15s
  external_labels:
    monitor: (MonitorPlus)
  scrape_interval: 15s
rule_files: []
scrape_configs:
- file_sd_configs:
  - files:
    - host.yml
  job_name: Host
  scrape_interval: 1m
- file_sd_configs:
  - files:
    - mysql.yml
  job_name: MySQL
  scrape_interval: 1m
- job_name: prometheus
  static_configs:
  - targets:
    - localhost:9090

[root@server prometheus]# cat mysql.yml
- targets:
  - 10.10.20.7:9104
  - 10.10.20.7:5555
```

@ITPUB博客

部署exporter端

对于mysql采集只需要配置连接信息，并启动mysql_exporter即可

```
[root@server prometheus]# export DATA_SOURCE_NAME=test:test@10.10.20.8:9306/"
[root@server prometheus]# ./usr/local/bin/mysql_exporter
INFO[0000] Starting mysql_exporter (version@9.9.0, branch=master, revision=8400af20ccdbf6b5e0faa2c925c56c48cd78d70b) source=mysql_exporter.go:432
INFO[0000] Build context (go=go1.6.3, user=root@c131c66ca20, date=20160926-18:28:09) source=mysql_exporter.go:433
INFO[0000] Listening on :9104 source=mysql_exporter.go:451
```

@ITPUB博客

完成配置之后即可通过mysql_exporter采集mysql性能指标

```
mysql_global_status_bytes_received 1.7309108e+07
# HELP mysql_global_status_bytes_sent Generic metric from SHOW GLOBAL STATUS.
# TYPE mysql_global_status_bytes_sent untyped
mysql_global_status_bytes_sent 2.4067284e+07
# HELP mysql_global_status_commands_total Total number of executed MySQL commands.
# TYPE mysql_global_status_commands_total counter
mysql_global_status_commands_total{command="admin_commands"} 0
mysql_global_status_commands_total{command="alter_db"} 0
mysql_global_status_commands_total{command="alter_db_upgrade"} 0
mysql_global_status_commands_total{command="alter_event"} 0
mysql_global_status_commands_total{command="alter_function"} 0
mysql_global_status_commands_total{command="alter_procedure"} 0
mysql_global_status_commands_total{command="alter_server"} 0
mysql_global_status_commands_total{command="alter_table"} 0
mysql_global_status_commands_total{command="alter_tablespace"} 0
mysql_global_status_commands_total{command="alter_user"} 0
mysql_global_status_commands_total{command="analyze"} 0
mysql_global_status_commands_total{command="begin"} 0
mysql_global_status_commands_total{command="binlog"} 0
mysql_global_status_commands_total{command="call_procedure"} 0
mysql_global_status_commands_total{command="change_db"} 0
mysql_global_status_commands_total{command="change_master"} 0
mysql_global_status_commands_total{command="change_repl_filter"} 0
mysql_global_status_commands_total{command="check"} 0
mysql_global_status_commands_total{command="checksum"} 0
```

@ITPUB博客

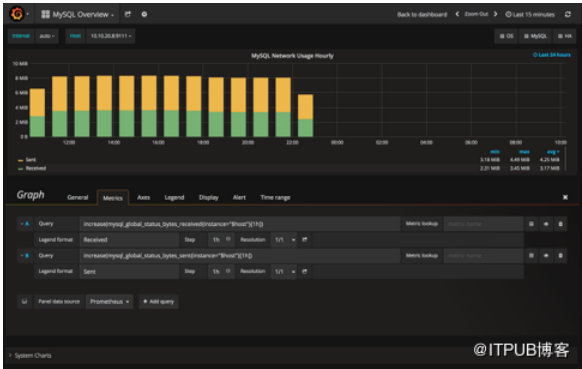
然后我们在prometheus服务端也可以查询到采集的mysql性能指标



@ITPUB博客

基于这些采集指标和Prometheus提供的规则计算语句，我们可以实现一些高纬度的查询需求，比如，
increase(mysql_global_status_bytes_received{instance="\$host"}[1h])

我们可以查询MySQL每小时接受到的字节数，然后将这个查询放到Grafana中，就可以展现非常酷炫的性能图表。



@ITPUB博客

而目前结合Prometheus和Grafana的MySQL监控方案已经有开源实现，我们很轻松可以搭建一套基于Prometheus的监控系统



对于告警方面我们也可以基于Prometheus丰富的查询语句实现复杂告警逻辑
比如我们要对MySQL备库进行监控，如果复制IO线程未运行或者复制SQL线程未运行并且持续2分钟就发送告警我们可以使用如下这条告警规则。

```
• # Alert: The replication IO or SQL threads are stopped.

• ALERT MySQLReplicationNotRunning

• IF mysql_slave_status_slave_io_running == 0 OR mysql_slave_status_slave_sql_running == 0

• FOR 2m

• LABELS {

• severity = "critical"

• }

• ANNOTATIONS {

• summary = "Slave replication is not running",

• description = "Slave replication (IO or SQL) has been down for more than 2 minutes.",
```

```
•  
    }
```

在比如，我们要监控MySQL备库延迟大于30秒并且预测在未来2分钟之后大于0秒持续1分钟，则告警

```
•  
    # Alert: The replicaion lag is non-zero and it predicted to not recover within
```

```
•  
    #      2 minutes. This allows for a small amount of replication lag.
```

```
•  
    ALERT MySQLReplicationLag
```

```
•  
    IF
```

```
•  
    (mysql_slave_lag_seconds > 30)
```

```
•  
    AND on (instance)
```

```
•  
    (predict_linear(mysql_slave_lag_seconds[5m], 60*2) > 0)
```

```
•  
    FOR 1m
```

```
•  
    LABELS {
```

```
•  
    severity = "critical"
```

```
•  
    }
```

```
•  
    ANNOTATIONS {
```

```
•
    summary = "MySQL slave replication is lagging",

•
    description = "The mysql slave replication has fallen behind and is not recovering",

•
}
```

当然在数据库方面不只是有MySQL的监控实现，目前业界也有很多其他开源实现，所以在数据库监控方面也能实现开箱即用的效果

- mysql_exporter https://github.com/prometheus/mysqld_exporter
- redis_exporter https://github.com/oliver006/redis_exporter
- postgres_exporter https://github.com/wrouesnel/postgres_exporter
- mongodb_exporter https://github.com/percona/mongodb_exporter

来自“ITPUB博客”，链接：<http://blog.itpub.net/28218939/viewspace-2137745/>，如需转载，请注明出处，否则将追究法律责任。


👍 点赞 | 0

☆ 收藏 | 0

分享到：

上一篇：[【MySQL】主从GTID复制修复](#)

下一篇：[【MySQL】load data语句详解（一）](#)



登录后发表评论

登录

全部评论

