PPTV
每个人的网络电视

分布式监控系统 Zabbix实践

姚仁捷
@超大杯摩卡星冰乐
2012.10.7

Update

- 2012.10.7: 增加了CMDB相关内容；增加了Puppet内容；增加rule_engine内容；Zabbix future。

Agenda

- What is Zabbix
  - Three Architecture of Zabbix
  - Item, Trigger, Action, Graph
  - Powerful zabbix_get
- Zabbix @ PPTV
  - Scalability
  - How we use it
    - Data Visualization
    - Development on Zabbix
    - With Puppet & CMDB
  - Architecture & Tuning
    - Zabbix Health
    - Architecture Change
    - Oracle Problems & Solutions
    - Configuration Parameters
    - Frontend Performance
- System to Platform
  - Zabbix Python API
  - Alert System, rule_engine
- Future
- Summary

名词解释1

- Host: 一台被监控的设备，服务器，交换机等。
- Item: 一个监控点，比如cpu load
- Trigger: 触发器（报警点），比如cpu load大于10。
- Action: trigger触发后的动作，比如发邮件
- Event: trigger每次状态（比如cpu load数据每次收到数据，都会针对trigger比较，并记录状态）
- Graph: 可以自定义的图表，不同item，坐标轴，图的类型等
- Screen: graph的集合，需要手动维护
- History: 每个item的每次获取的数据
- Trend: 针对每个item，每小时的min，max，avg值。

名词解释2

- Template: 模板，类似host，上面可以有item, trigger, action等。host如果属于某个template，那么template上的所有内容会“复制”到host上。
- Zabbix Server: Zabbix中央服务器
- Zabbix Proxy: Zabbix代理，分布式中使用
- VPS: Value Per Second，每秒处理数据量。这个值从数据库计算而出。可以定量的衡量server的压力。

WHAT IS ZABBIX

简介

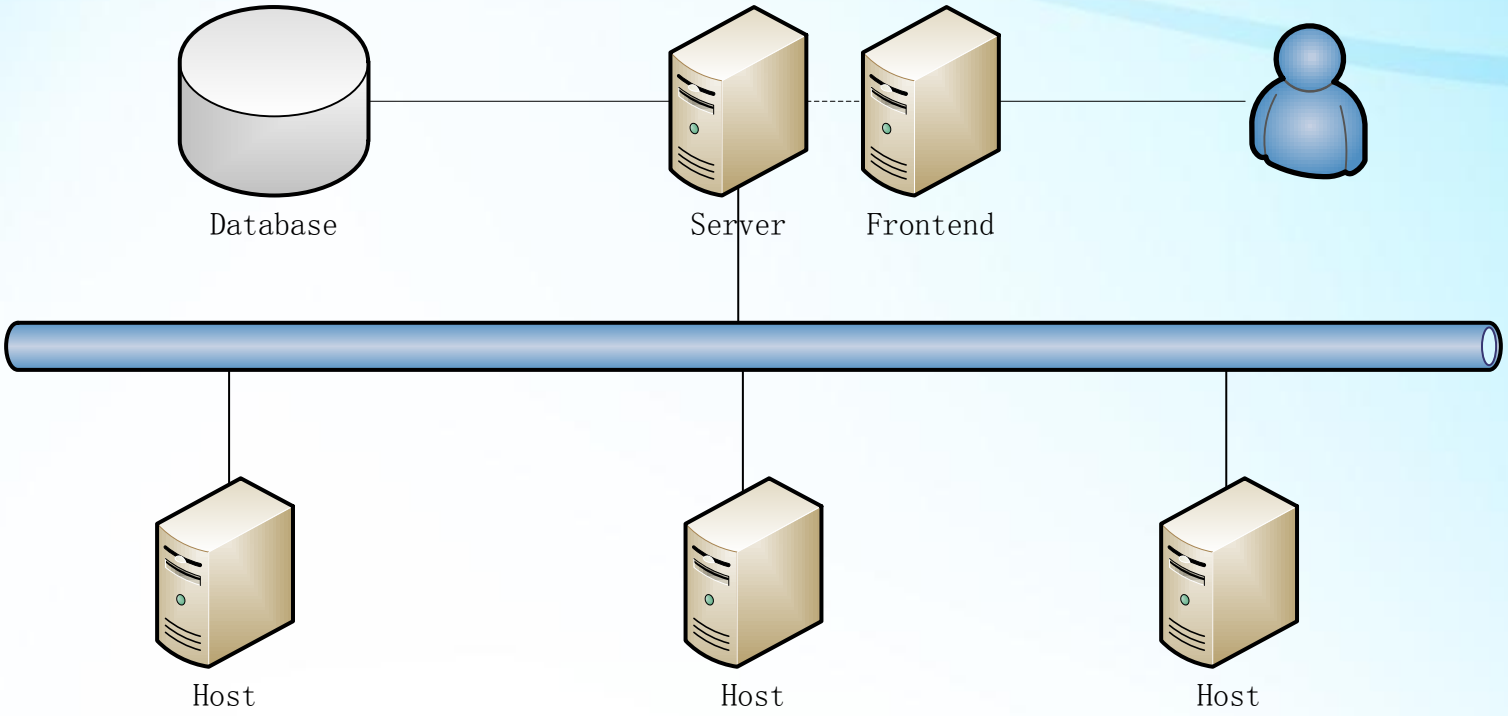
Zabbix是一个分布式监控系统，可以收集数据，展现数据，报警，存储数据。为了收集数据，它需要在监控的host上装一个agent（如果是简单的server到host的单方向监控，是不需要agent的）。Backend有数据库可以做数据分析。相对Nagios，Zabbix有强大（但性能烂，有时有问题）的前端，可以方便的配置。

三种架构

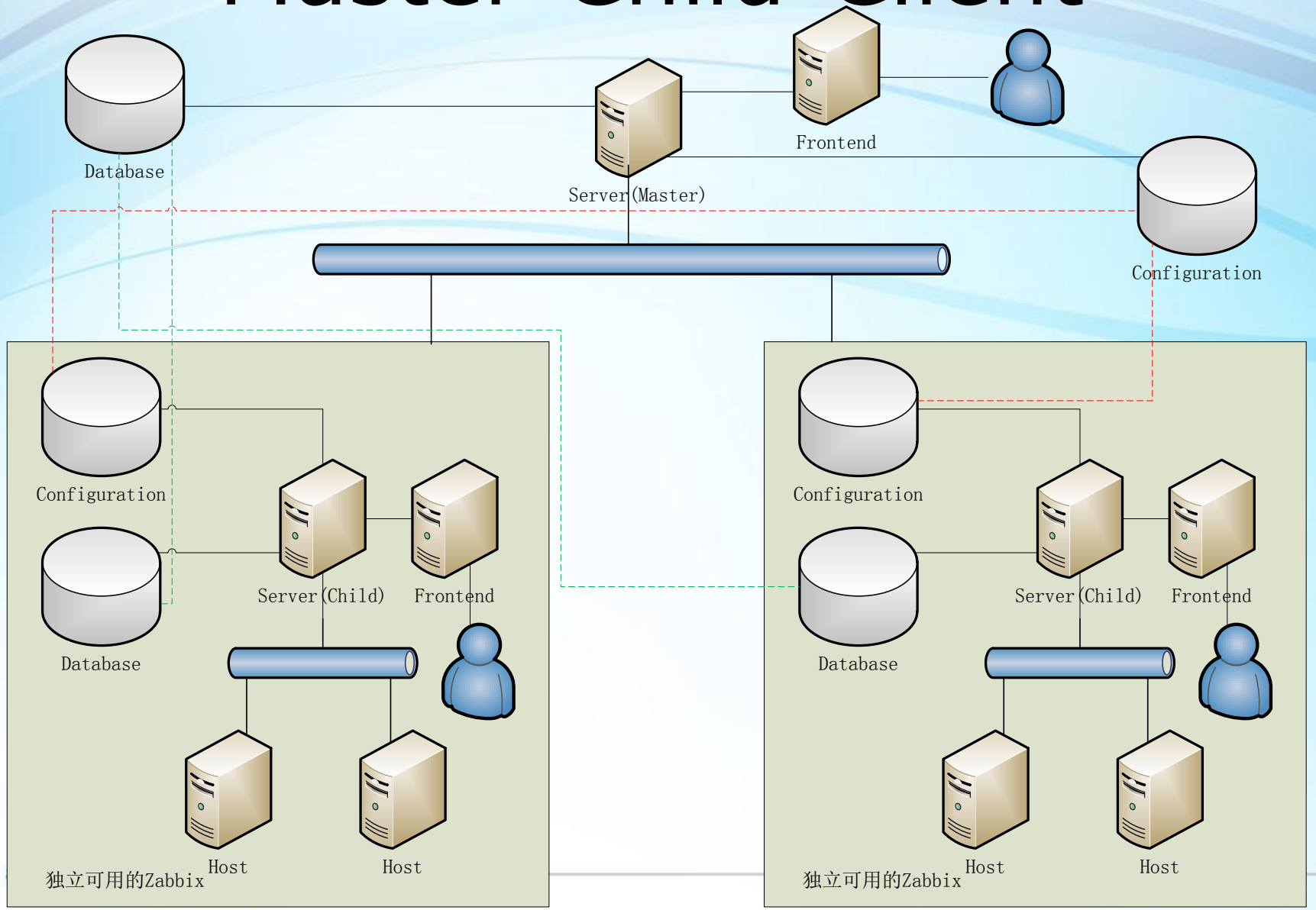
- Server-Client
- Master-Node-Client
- Server-Proxy-Client

Server-Client

- 最简单的实现方式，适合网络状况好，host少（具体数量看server配置）的case。



Master-Child-Client



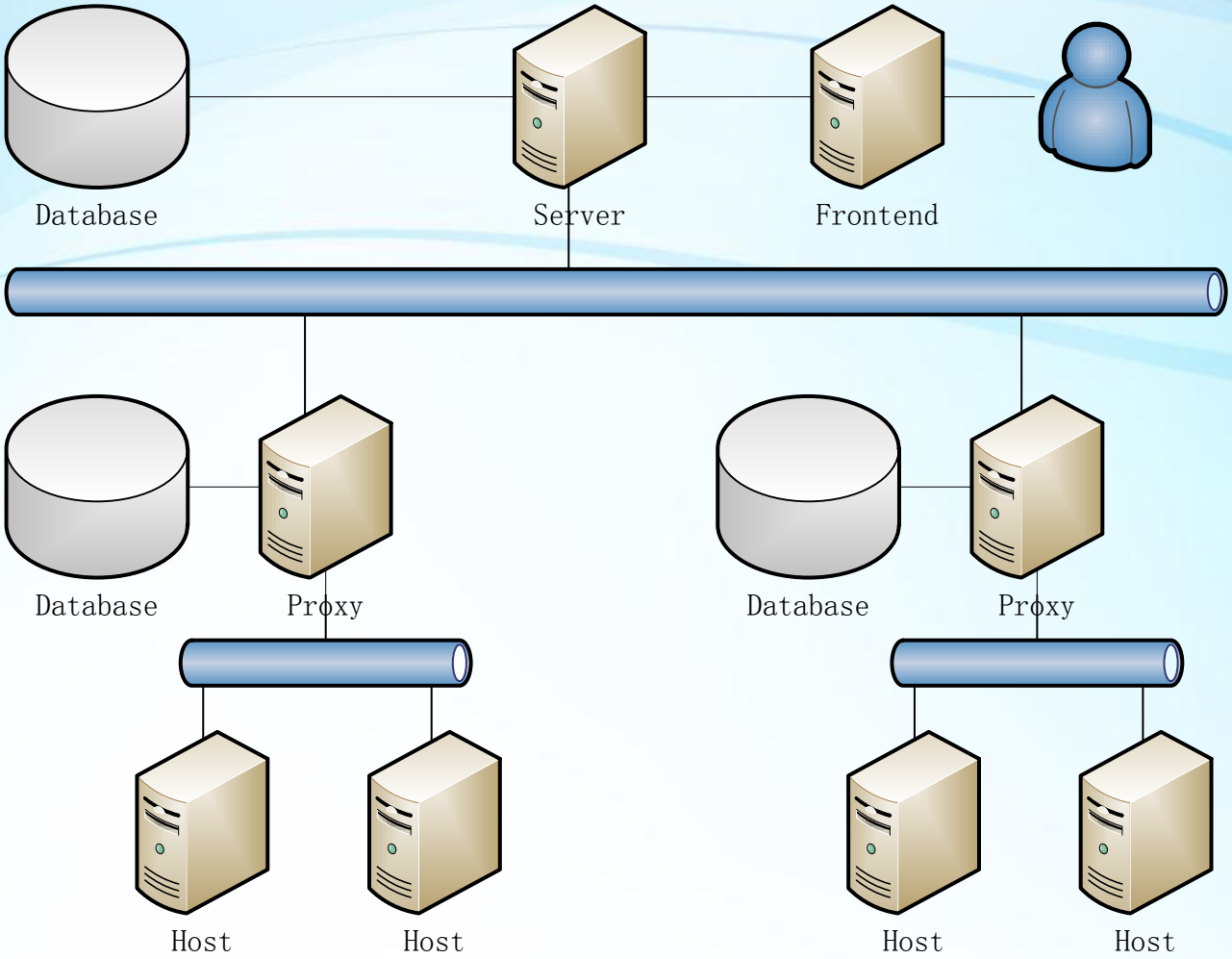
特性1

- Child-Client实际上就是前面说的Server-Client架构。加上Master，是解决host多了以后一台Server抗不住，用多个instance的解决方案。
- 每个instance是独立的一套zabbix，有数据库，前端（可选）。
- 热插拔，Child和Master的连接可以随时断开，不影响Child任何数据。

特性2

- Child定时给Master发送configuration，history，event。
- Master定时给Child发送configuration。
- 所有配置变更只能在child节点操作，不能再master操作。http://www.zabbix.com/forum/showthread.php?t=20863
- 支持树状结构，Child又可以是个Master。

Server-Proxy-Client



特性

- Proxy不会向Server同步Configuration，只会接收。
- Proxy的数据库定时会将数据传送给Server，Proxy本地数据库只保存最近没有发送的数据。

灵活的监控

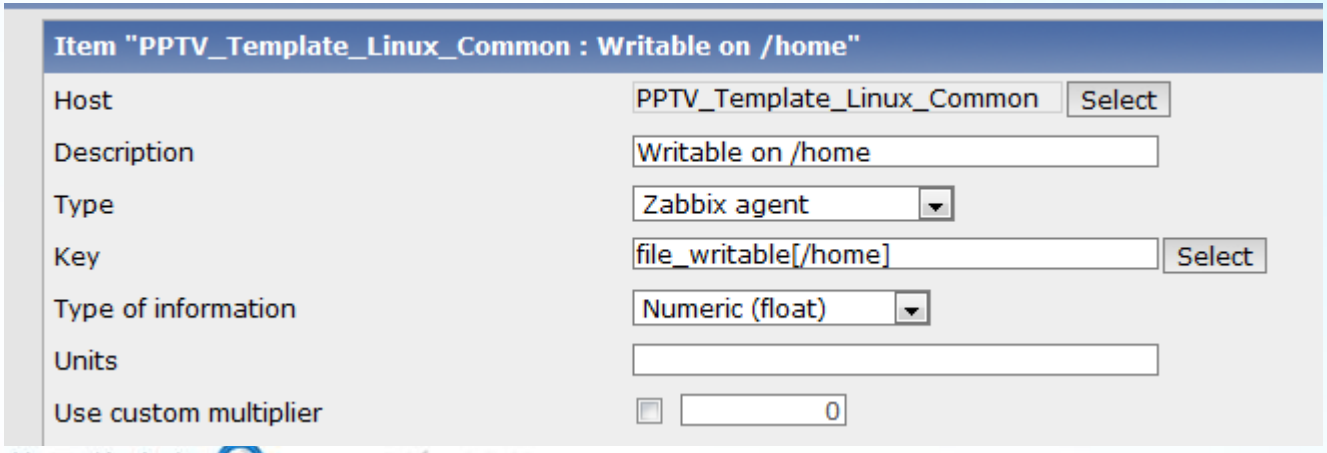
- 在监控的内容这方面，zabbix可以使用自定义的脚本，只需要脚本print出需要的值即可。简单来说，可以将其理解能获取结果的远程crontab。
- Agent的debug日志很详细，监控脚本如果有问题可以快速定位。

Item

- 一个Item（监控点）最重要的是key，即监控的内容。
- Agent运行时，会将本地的conf文件载入内存。
- Conf文件中配置了key，并指定这个key跑什么脚本。

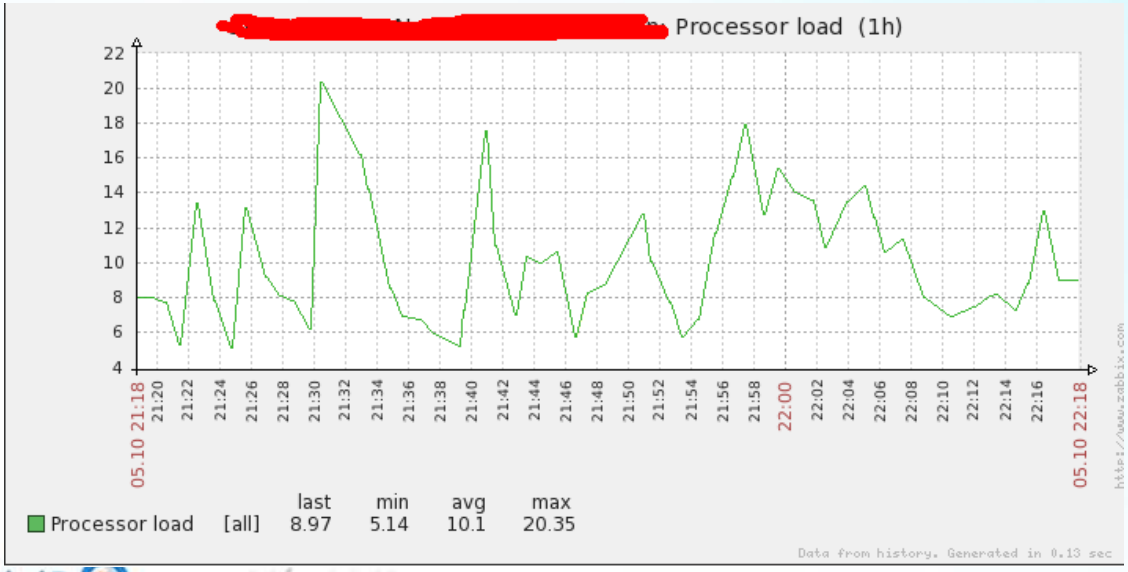
【例】/home只读监控

- Conf文件：UserParameter=file_writable[*],/usr/local/zabbix/external-script/file_writable.sh $1
- Key配置：



监控图方便外部引用

- 要使用下图，只要引用url：http://zabbix.test.cn/chart.php?itemid=1377787&period=3600&stime=20121005211843



Tips

- 如果脚本不需要参数，那么conf文件中不需要后面的[*]
- 修改conf文件后，要重启agent才能起效。
- 修改脚本不需要重启agent。

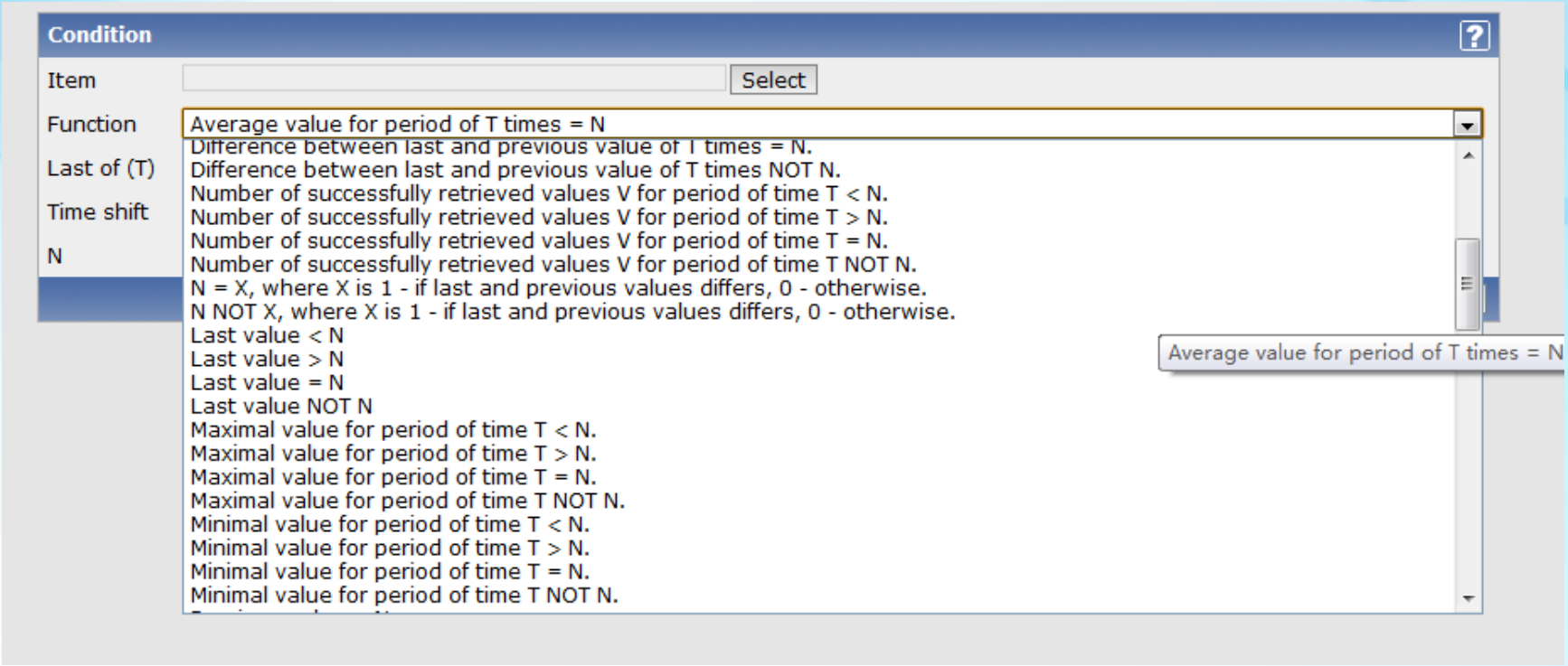
Trigger

- 一个Trigger（报警点），是报警的来源，即这次数据是否要报警。
- Zabbix支持非常多的报警条件，而且可以用and, or等连接。
- Trigger有Unknown，OK，Problem三种状态。

Trigger is Unknown ?

- 比如说一个Trigger是cpu load>10 and free memory < 100。那么当cpu load没取到值时，这个trigger的状态时unknown，因为Zabbix无法判断到底有没有问题。
- 注意：从unknown到problem状态时不会触发action的。只有从ok到problem才会触发，一直problem也不会重复触发。

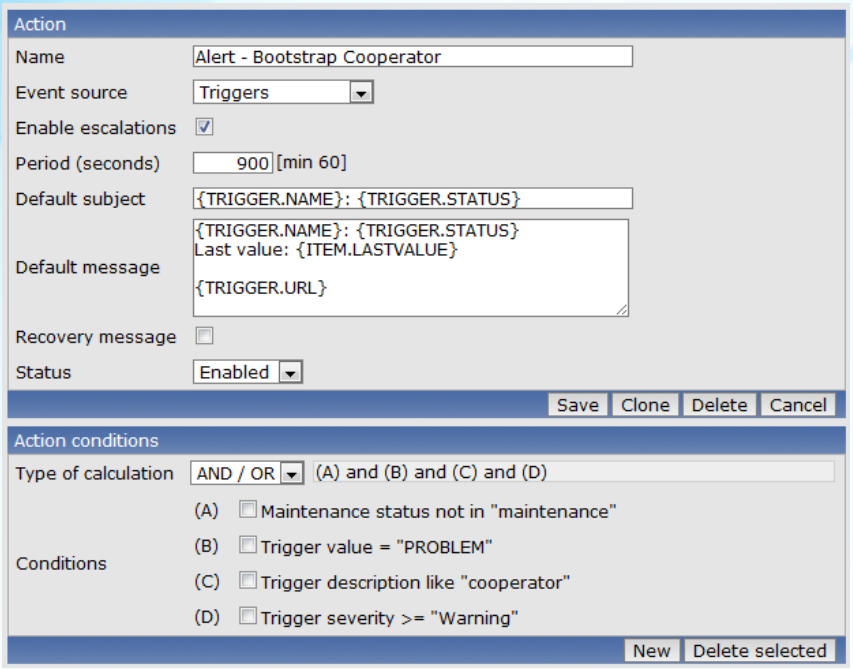
Trigger的各种逻辑



Action

- Trigger认为有问题，那么会触发action。
- Action可以配置发邮件等功能。
- 强大的Escalation功能。15分钟没有解决则发送给XXX，或者说30分钟没解决就继续发报警，这些都能通过escalation机制完成。

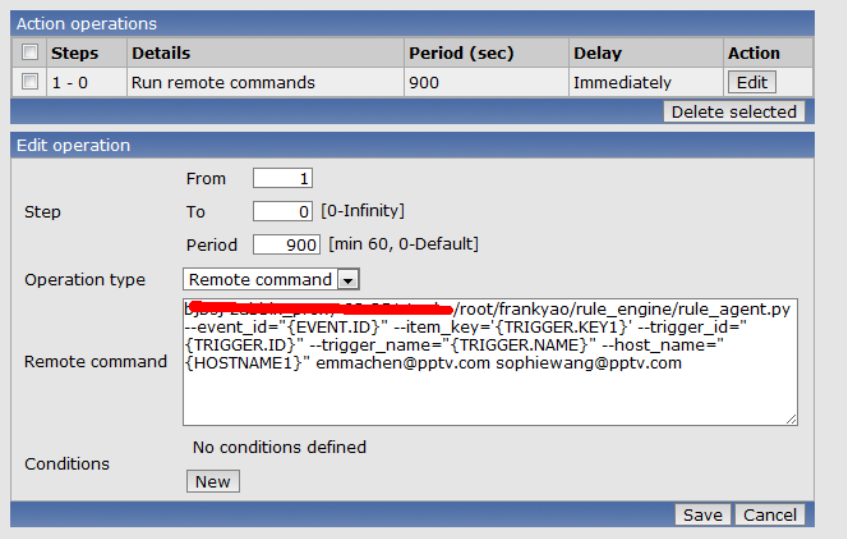
一个例子



- 可以看到action支持各种规则，比如这个例子中是规定trigger名字类似“cooperator”才会使用这个action

一个例子

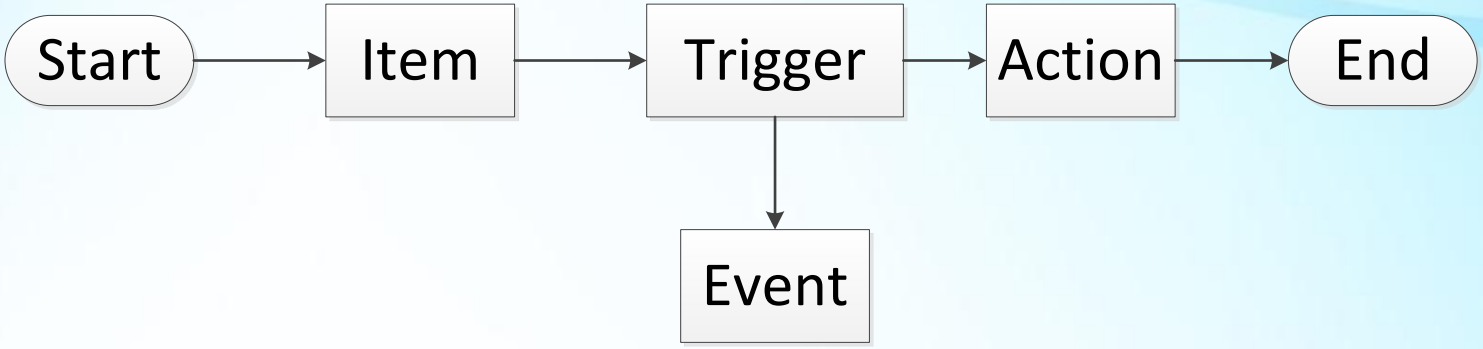
- Action的动作是使某一台机器执行命令。



Event

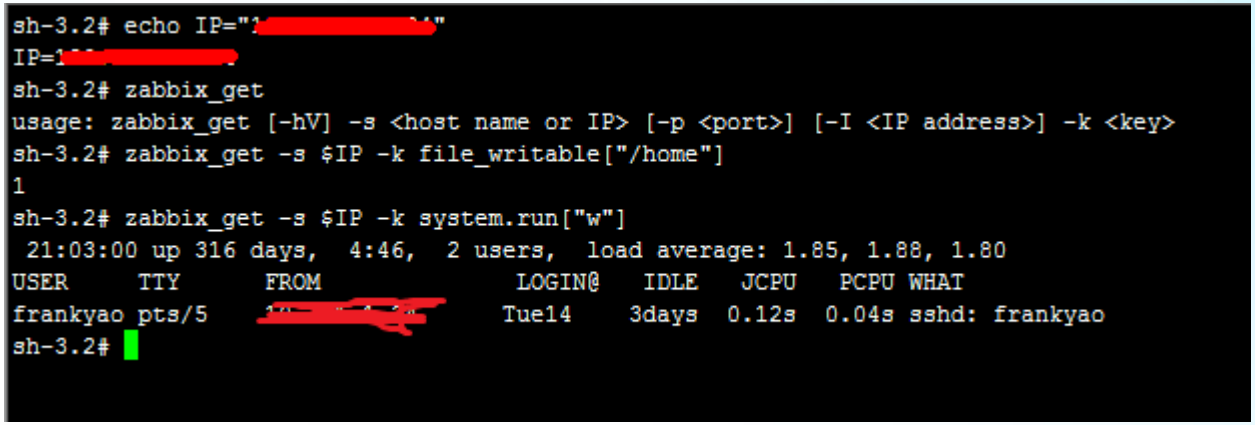
- Event表示一个事件，每次trigger判断为状态后，都会生成一个event（无论是ok，unknown还是problem）。

报警的流程



zabbix_get

- Zabbix_get是zabbix自带的用于远程获取item数据。由于自带的item有个叫“system.run”的，用于跑命令。所以zabbix_get有远程跑命令的功能。



So dangerous

- zabbix_get太过强大，完全可以写个脚本所有服务器跑个rm –rf /。
- 幸好zabbix也考虑到了这一点。Agent的配置中写了可以zabbix_get本机的白名单。

```
### Option: Server
#       List of comma delimited IP addresses (or hostnames) of Zabbix servers.
#       No spaces allowed. First entry is used for receiving list of and sending active checks.
#       If IPv6 support is enabled then '127.0.0.1', '::127.0.0.1', '::ffff:127.0.0.1' are treated equally.
#
# Mandatory: yes
# Default:
# Server=
```

Tips

- 假设A机器上跑zabbix_get，获取B机器的数据。B机器conf里写的Server写A的所有IP有时也无法zabbix_get。这时需要将A的出口IP写进去。如何获取出口IP？A上跑`curl ifconfig.me`就行了。

ZABBIX @ PPTV

Zabbix @ PPTV

SCALABILITY

Scalability

- 我们使用的Server-Proxy-Client架构。数据库是Oracle。
- Zabbix目前监控了73w个监控点，20w个报警点。貌似没见过更大规模的了☺。

STATUS OF ZABBIX

REPORT

Parameter	Value	Details
Zabbix server is running	Yes	[redacted]:10051
Number of hosts (monitored/not monitored/templates)	[redacted]	[redacted]
Number of items (monitored/disabled/not supported)	732726	552683 / 146727 / 33316
Number of triggers (enabled/disabled)[problem/unknown/ok]	197599	124019 / 73580 [108 / 32791 / 91118]
Number of users (online)	92	6
Required server performance, new values per second	1779.86	-

Updated: 21:40:51

Zabbix @ PPTV

HOW WE USE IT

Section Agenda

- Work with Puppet
- Host分组
- 监控了哪些数据
- 监控数据的查看
- 二次开发
- 添加监控
- 数据分析

With Puppet

- Zabbix监控的host需要同步监控脚本，配置文件等。不可能一台一台机器ssh上去cp文件。PPTV的infrastructure中Puppet帮助Zabbix完成了这个工作。它会同步Zabbix的目录，保持zabbix_agent的正常运行。
- 个人建议Zabbix一定要和Puppet（或类似）的系统一起运作，会极大的减低运维的成本。

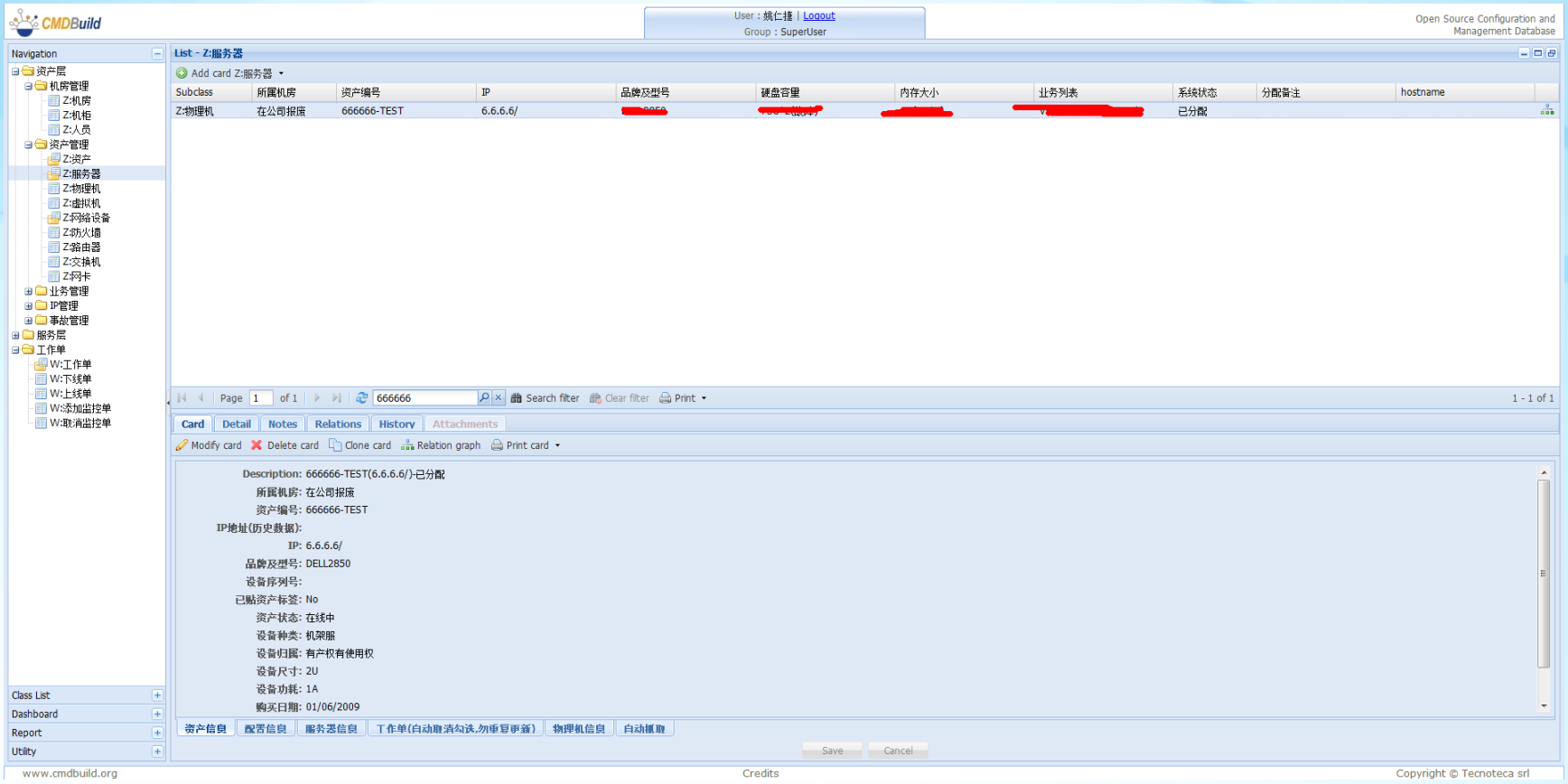
Puppet Module of Zabbix

```
file { "/usr/local/zabbix/":
          mode => 0755, owner => zabbix, group => zabbix,
          source => $architecture  ? {
                    x86_64 => "puppet://$fileserver/files/zabbix/x86_64" ,
                    i386 => "puppet://$fileserver/files/zabbix/i386" ,
           },
          recurse => true,
           force => true,
           purge => true,
          ignore => File[ "external-script","external-conf","external-lib","zabbix_agentd.pid" ,"crow"] ,
}
file { "/usr/local/zabbix/external-script":
          mode => 0755, owner => zabbix, group => zabbix,
          source => "puppet://$fileserver/files/zabbix/external-script" ,
          require => File["/usr/local/zabbix/"],
          recurse => true,
          force => true,
          purge => true,
}
file { "/usr/local/zabbix/crow":
          mode => 0755, owner => zabbix, group => zabbix,
          source => "puppet://$fileserver/files/zabbix/crow" ,
          require => File["/usr/local/zabbix/"],
          recurse => true,
```

With CMDB

- 现在ops的配置信息都是以CMDB为准。现在Zabbix的很多功能都依赖于CMDB。
- 具体关于CMDB的实现，要@海阳的阳 才能说清楚了

CMDB界面


CMDBuild
User : 桃仁捷 | Logout
Group : SuperUser
Open Source Configuration and Management Database
Navigation
资产层
机房管理
Z:机房
Z:机柜
Z:人员
资产管理
Z:资产
Z:服务器
Z:物理机
Z:虚拟机
Z:网络设备
Z:防火墙
Z:路由器
Z:交换机
Z:网卡
业务管理
IP管理
事故管理
服务层
工作单
W:工作单
W:下线单
W:上线单
W:添加监控单
W:取消监控单
Class List
Dashboard
Report
Utility
List - Z:服务器
Add card Z:服务器
Subclass
所属机房
资产编号
IP
品牌及型号
硬盘容量
内存大小
业务列表
系统状态
分配备注
hostname
Z:物理机
在公司报废
666666-TEST
6.6.6.6/
已分配
Page 1 of 1
666666
Search filter
Clear filter
Print
1 - 1 of 1
Card
Detail
Notes
Relations
History
Attachments
Modify card
Delete card
Clone card
Relation graph
Print card
Description: 666666-TEST(6.6.6.6/)-已分配
所属机房: 在公司报废
资产编号: 666666-TEST
IP地址(历史数据):
IP: 6.6.6.6/
品牌及型号: DELL2850
设备序列号:
已贴资产标签: No
资产状态: 在线中
设备种类: 机架服
设备归属: 有产权有使用权
设备尺寸: 2U
设备功耗: 1A
购买日期: 01/06/2009
资产信息
配置信息
服务器信息
工作单(自动取消勾选,勿重复更新)
物理机信息
自动抓取
Save
Cancel
www.cmdbuild.org
Credits
Copyright © Tecnoteca srl

Work with CMDB

- Zabbix添加监控与CMDB集成。人的操作总会有纰漏，需要严格的执行流程。Zabbix中添加监控实际就是一台机器的上线，动作的发起者应该是CMDB。
- CMDB中对机器有“系统状态”属性。如果机器在维护中，那么不应该发送报警，这样会浪费值班同学的时间。
- Screen与CMDB中关系的集成。以业务的视角来看一个系统。

Host分组

- Zabbix有Group的概念，可以将一些Host放在一个Group中。
- 我们根据Host的hostname会以三个维度分组。
  - Location：地域（CDT）
  - Module：业务（VOD）
  - Function：跑了什么服务（Resin）

监控数据类型

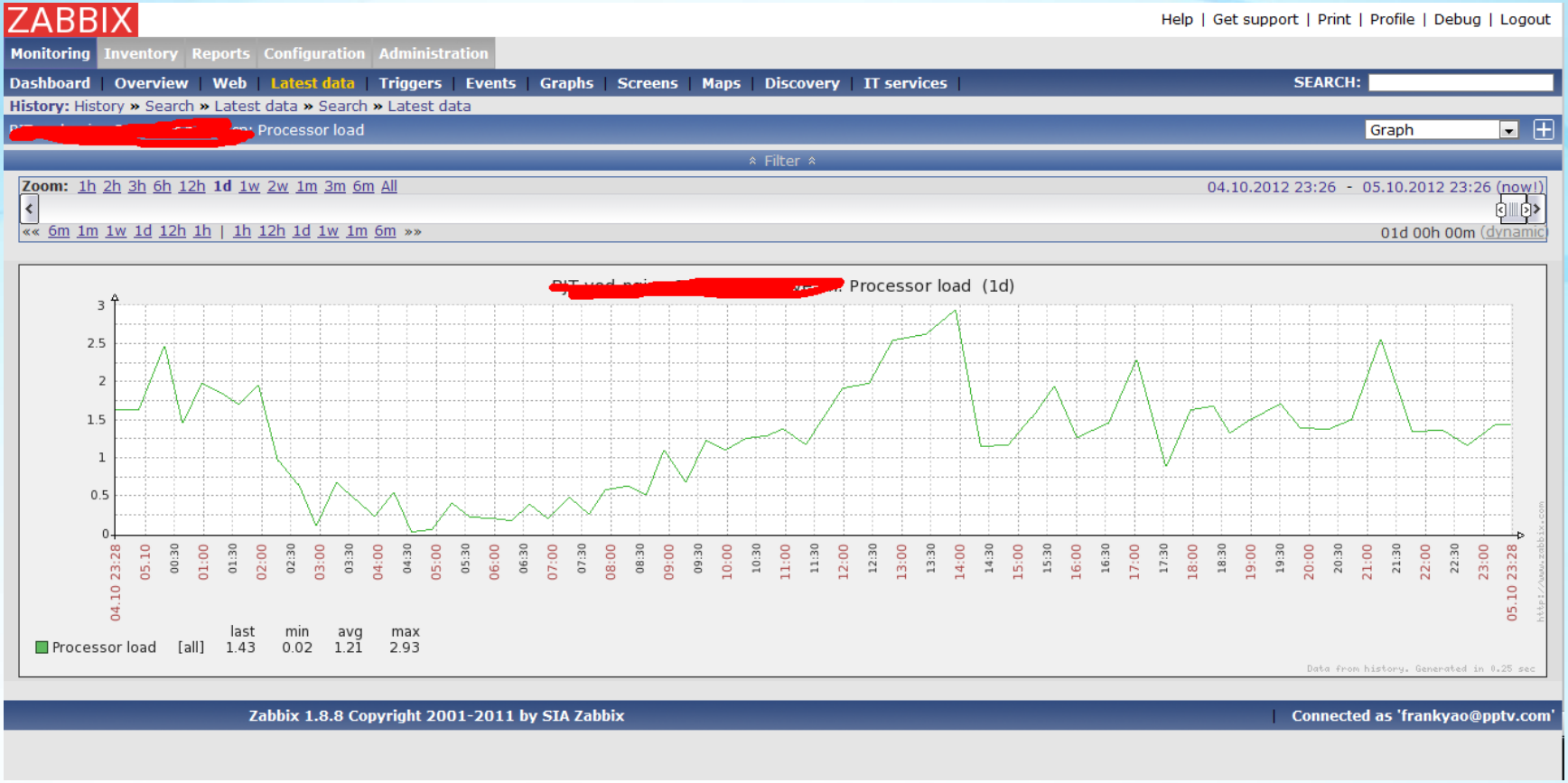
- 现在主要有三种监控数据：
  - 基础监控：如cpu，memory等。
  - 应用监控：resin，mysql等。
  - LB日志监控：Flume为数据源。
  - 运营数据监控：流量，注册人数等。

查看监控数据

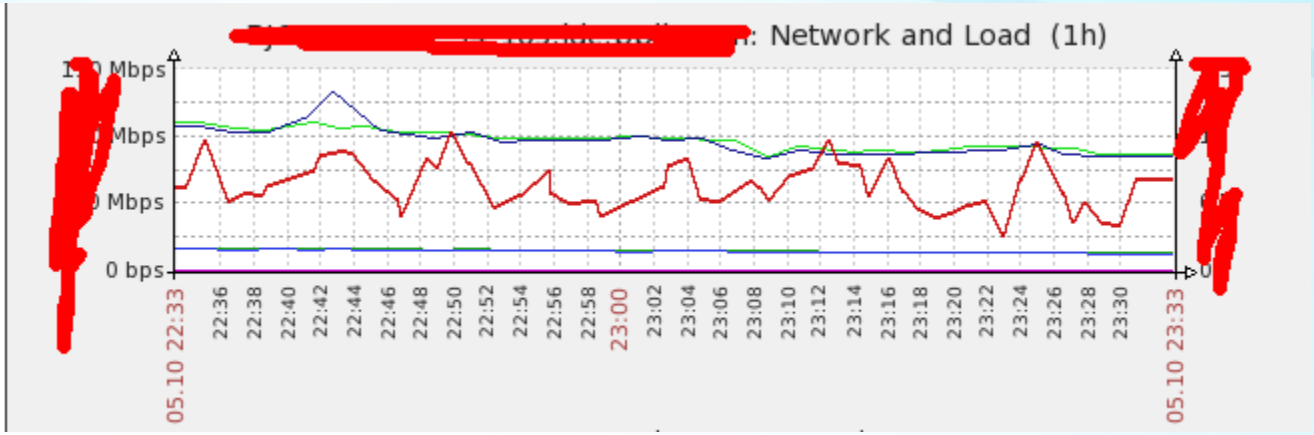
- Troubleshooting时肯定要看监控数据。
- 下面是一个host的latest data。

Description	Last check	Last value	Change	History
Agent (2 Items)				
Business Bandwith (6 Items)				
CPU (4 Items)				
CPU idle time (avg1)	05 Oct 2012 23:25:44	91.78	+1.96	Graph
CPU system time (avg1)	05 Oct 2012 23:08:38	0.45	-0.2	Graph
CPU Numbers	05 Oct 2012 12:08:35	16	-	Graph
Processor load	05 Oct 2012 23:14:02	1.43	+0.26	Graph
Disk Health (3 Items)				
Disk_Performance (11 Items)				
Download Speed QoS (3 Items)				
Filesystem (32 Items)				
General (3 Items)				
Integrity (1 Items)				
Memory (2 Items)				
Netcard Max Speed (2 Items)				
Network (4 Items)				
Nginx (1 Items)				
Nginx_Log (50 Items)				

CPU Load图


ZABBIX
Help | Get support | Print | Profile | Debug | Logout
Monitoring | Inventory | Reports | Configuration | Administration
Dashboard | Overview | Web | Latest data | Triggers | Events | Graphs | Screens | Maps | Discovery | IT services
SEARCH:
History: History » Search » Latest data » Search » Latest data
Processor load
Graph
Filter
Zoom: 1h 2h 3h 6h 12h 1d 1w 2w 1m 3m 6m All
04.10.2012 23:26 - 05.10.2012 23:26 (now!)
«« 6m 1m 1w 1d 12h 1h | 1h 12h 1d 1w 1m 6m »»
01d 00h 00m (dynamic)
Processor load (1d)
3
2.5
2
1.5
1
0.5
0
04.10 23:28
05.10 23:28
last min avg max
Processor load [all] 1.43 0.02 1.21 2.93
Data from history. Generated in 0.25 sec
Zabbix 1.8.8 Copyright 2001-2011 by SIA Zabbix
Connected as 'frankyao@pptv.com'

多个数据


[redacted]: Network and Load (1h)
Mbps
Mbps
Mbps
0 bps
22:33
22:36
22:38
22:40
22:42
22:44
22:46
22:48
22:50
22:52
22:54
22:56
22:58
23:00
23:02
23:04
23:06
23:08
23:10
23:12
23:14
23:16
23:18
23:20
23:22
23:24
23:26
23:28
23:30
05.10
05.10 23:33

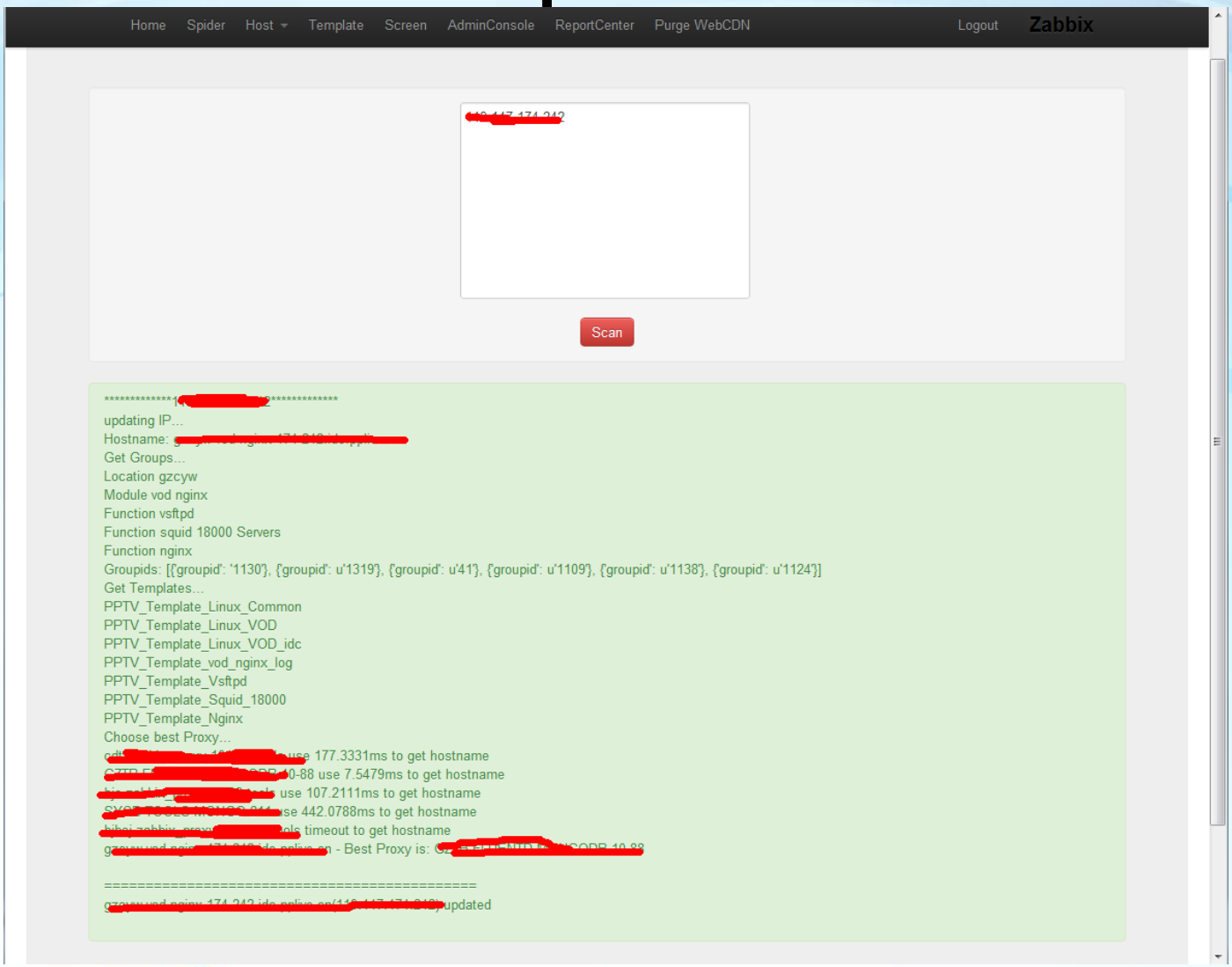
二次开发

- 针对performance问题对Zabbix代码做了Patch。提交的patch被reject了。所以我们现在Zabbix一直是1.8.8版本。Release notes没有什么值得更新的地方，1.8.8足够。
- 傻瓜化添加监控，只需要ip即可，自动添加对应服务监控。
- 针对需求，修改了一些php代码，并与CAS集成。
- Screen功能迁出，全部重新开发。
- 报警系统重新开发。
- 从apache迁移到nginx+php-fpm。
- 写了Zabbix Python API。

添加监控

- 监控系统最麻烦的就是加监控了。这个机器是上跑了resin，那么要加resin的监控；跑了mysql要加mysql监控。
- db机器load多少才报警，web机器load多少才报警都是不同的。手动加是非常麻烦的，而且不现实的。
- 我们使用Spider来添加机器的监控，只需要输入IP，自动会根据机器上的业务添加监控。

Spider


Home
Spider
Host
Template
Screen
AdminConsole
ReportCenter
Purge WebCDN
Logout
Zabbix
Scan
updating IP...
Get Groups...
Location gzcyw
Module vod nginx
Function vsftpd
Function squid 18000 Servers
Function nginx
Groupids: [{'groupid': '1130'}, {'groupid': u'1319'}, {'groupid': u'41'}, {'groupid': u'1109'}, {'groupid': u'1138'}, {'groupid': u'1124'}]
Get Templates...
PPTV_Template_Linux_Common
PPTV_Template_Linux_VOD
PPTV_Template_Linux_VOD_idc
PPTV_Template_vod_nginx_log
PPTV_Template_Vsftpd
PPTV_Template_Squid_18000
PPTV_Template_Nginx
Choose best Proxy...
use 177.3331ms to get hostname
use 7.5479ms to get hostname
use 107.2111ms to get hostname
use 442.0788ms to get hostname
timeout to get hostname
- Best Proxy is:
==========
updated

Admin Console

- Web上直接执行命令。用户和命令都有白名单，黑名单。

Screen

Home Spider Host Template Screen AdminConsole ReportCenter Purge WebCDN Logout Zabbix

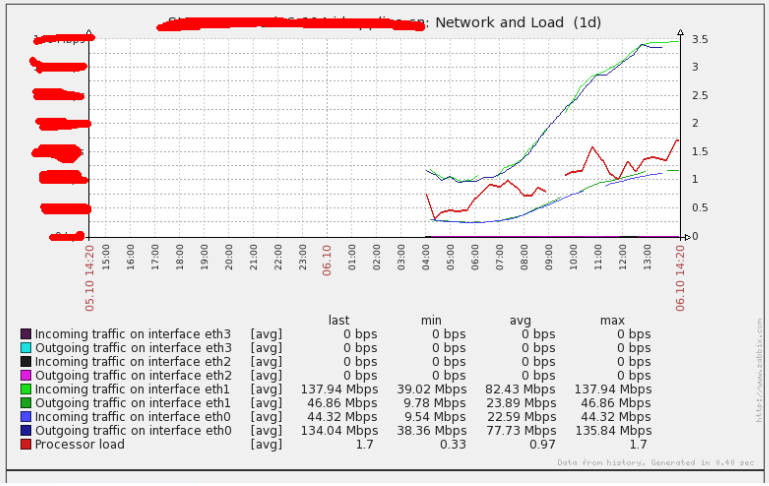
VOD WEB WEB_CMDB2 Database ShowTime CDN WEB_CMDB1 HADOOP Traffic P2P DAC-log_status Mainsite IDC Zabbix Origin Screen

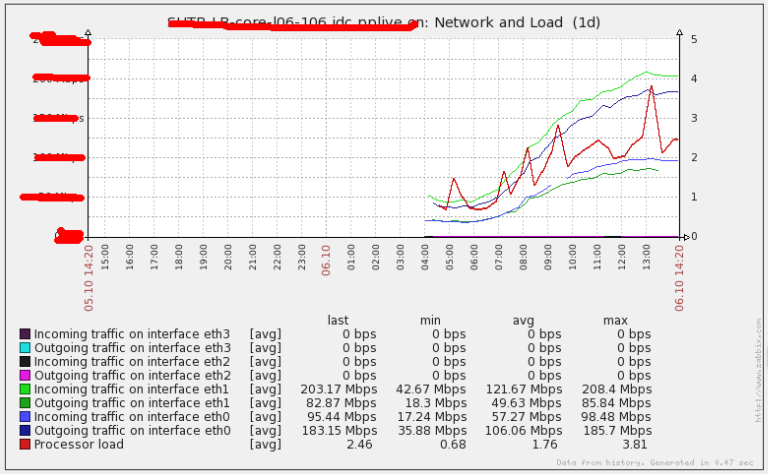
SH Network and Load

Owner: Qijun Wu

Start Time 2012/10/5-14:16:1 End Time 2012/10/6-14:16:1

Month Week AutoRefresh Go!





数据分析

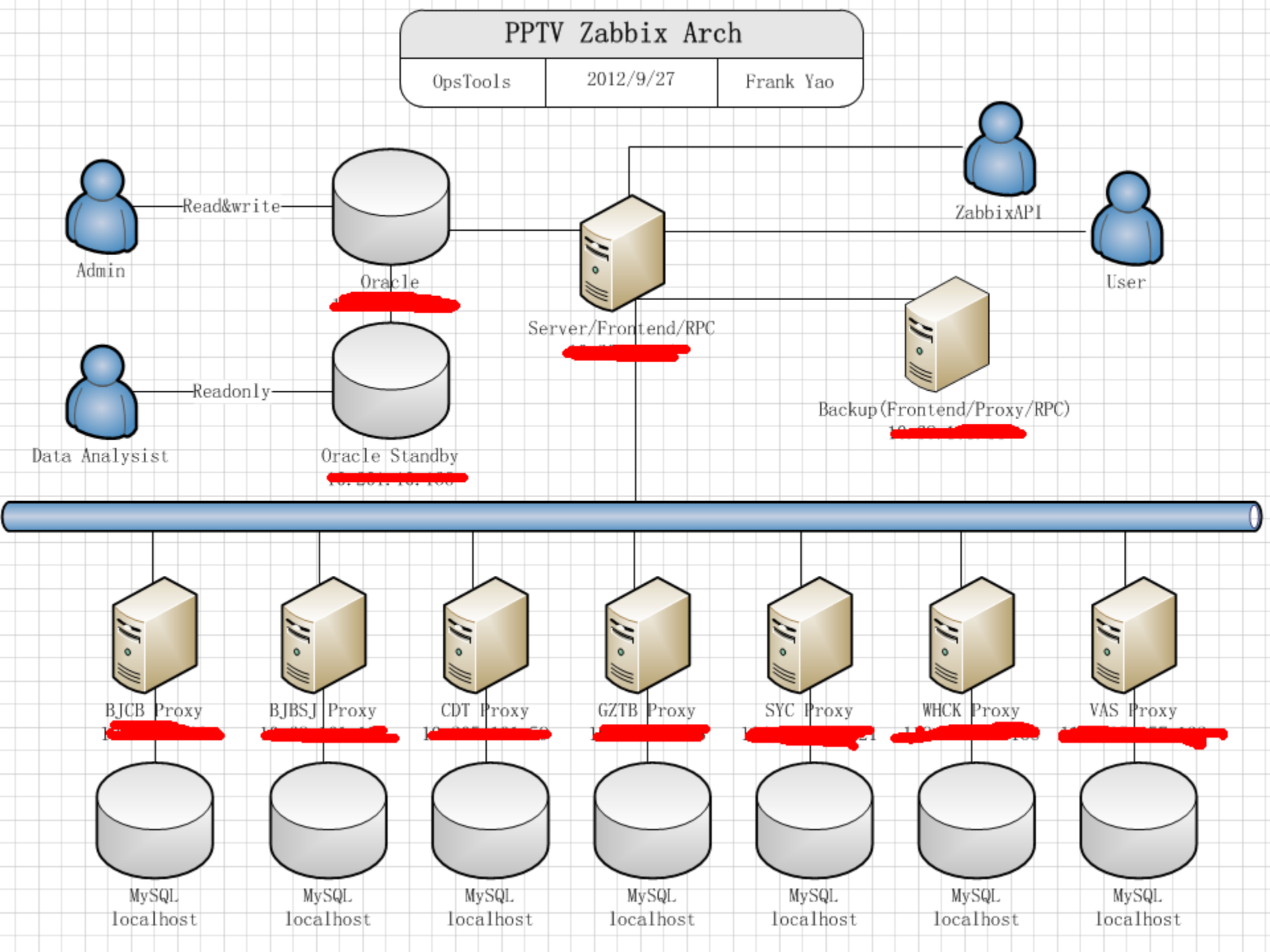
- 由于使用了Server-Proxy-Client架构，所有的数据都在Server的backend数据库中。数据库中有非常多的监控数据可供分析，做report，capacity review等。
- 我们有一台Oracle主库和一台standby。Standby开放readonly权限，大家都可以在上面做数据分析。主库则由于非常critical，只有tools team和ops一些人员才有admin权限。

Zabbix @ PPTV

ARCHITECTURE & TUNING

Architecture

- 我们使用的是Server-Proxy-Client架构。1台Server，7台Proxy，1台Oracle主库，1台Oracle standby，1台Backup Proxy。每台Proxy有自己的本地MySQL。Frontend、RPC在Server这台机器上，同时在Backup Proxy上有两者的backup。
- 目前的架构图见下页。


PPTV Zabbix Arch
OpsTools
2012/9/27
Frank Yao
Admin
Read&write
Oracle
ZabbixAPI
User
Server/Frontend/RPC
Backup(Frontend/Proxy/RPC)
Data Analysist
Readonly
Oracle Standby
BJCB Proxy
BJBSJ Proxy
CDT Proxy
GZTB Proxy
SYC Proxy
WHCK Proxy
VAS Proxy
MySQL
localhost

Zabbix Health

- 怎样衡量一个Zabbix系统（Server，Proxy）是否健康呢？这里，推荐一种方法，是统计最近一小时内，数据更新的百分比——Update Percent。
- SQL中的key_是为了排除一些会影响计算的item。

```
select a.aa/b.bb from
    (select count(*) as aa from items
        where lastclock > :1 and delay < 900 and status = 0
        and hostid in (select hostid from hosts where proxy_hostid=:2 and status=0)
        and key_ not like '%nginx%' and key_ not like '%checkURL%' and key_ not like 'lbdetail%') a,
    (select count(*) as bb from items
        where delay < 900 and status = 0
        and hostid in (select hostid from hosts where proxy_hostid=:2 and status=0)
        and key_ not like '%nginx%' and key_ not like '%checkURL%' and key_ not like 'lbdetail%') b
```

Nagios on Zabbix

- Zabbix是监控系统，那么Zabbix出问题了如何监控呢？我们使用了非常简单的Nagios来监控Zabbix。
- 后来，Nagios逐渐扩展到监控所有Tools的关键指标。

Zabbix Health

```
[t    le@zabbix-dev tool    check]$ ./update_percent_proxy.py
B TBSI Zabbix Proxy 101.44: 88.99, max lastclock: Sat Oct  6 14:33:43 2012
hibei zabbix proxy 101.109 8081: 69.03, max lastclock: Sat Oct  6 14:33:45 2012
cd  zabbix      : 82.16, max lastclock: Sat Oct  6 14:34:07 2012
SWCD TOOLS MOVSO 211: 60.03, max lastclock: Sat Oct  6 14:33:43 2012
whok zabbix-proxy-18      : 63.76, max lastclock: Sat Oct  6 14:33:42 2012
B  Zabbix 11: 75.57, max lastclock: Sat Oct  6 14:34:41 2012
```

[redacted]	Current Load	OK	10-06-2012 14:32:27	15d 20h 23m 58s	1/4	OK - load average: 0.00, 0.02, 0.00
	Current Users	OK	10-06-2012 14:32:27	229d 19h 54m 19s	1/4	USERS OK - 6 users currently logged in
	Haproxy	OK	10-06-2012 14:32:27	39d 0h 29m 48s	1/4	TCP OK - 0.000 second response time on port 8443
	Mysql	OK	10-06-2012 14:35:30	10d 5h 5m 57s	1/4	Uptime: 961685 Threads: 4 Questions: 752489 Slow queries: 0 Opens: 50 Flush tables: 1 Open tables: 43 Queries per second avg: 0.782
	Root Partition	OK	10-06-2012 14:32:27	229d 19h 56m 49s	1/4	DISK OK - free space: / 14688 MB (78% inode=97%):
	Swap Usage	OK	10-06-2012 14:32:27	109d 2h 19m 44s	1/4	SWAP OK - 100% free (4094 MB out of 4094 MB)
	Total Processes	OK	10-06-2012 14:34:24	8d 19h 27m 0s	1/4	PROCS OK: 69 processes with STATE = RSZDT
	event console loader log check	OK	10-06-2012 14:32:27	18d 0h 28m 57s	1/1	(null)
	rule_engine_queue_consumer	OK	10-06-2012 14:35:30	5d 20h 55m 56s	1/4	kenlovefrank ConsumerCount: 1.0
	rule_engine_queue_memoryUsage	OK	10-06-2012 14:32:35	34d 22h 2m 18s	1/4	kenlovefrank MemoryPercentUsage: 0.0
	zabbix_events_queue_consumer	OK	10-06-2012 14:32:35	57d 22h 13m 8s	1/4	zabbix_events ConsumerCount: 1.0
	zabbix_events_queue_memoryUsage	OK	10-06-2012 14:32:35	156d 3h 15m 25s	1/4	zabbix_events MemoryPercentUsage: 0.0
zabbix_db1	swap free percent	OK	10-06-2012 14:35:30	0d 3h 25m 54s	1/4	zabbix agent key 94.593537: system.swap.size[,pfree]
	zabbix agent	OK	10-06-2012 14:31:30	7d 1h 4m 55s	1/4	TCP OK - 0.048 second response time on port 10050
zabbix_db2	swap free percent	OK	10-06-2012 14:32:29	25d 4h 33m 58s	1/4	zabbix agent key 86.238786: system.swap.size[,pfree]
	zabbix agent	OK	10-06-2012 14:34:27	7d 21h 41m 57s	1/4	TCP OK - 0.039 second response time on port 10050
zabbix_master	Zabbix server process	OK	10-06-2012 14:32:33	3d 20h 23m 51s	1/4	zabbix agent key 136: proc.num[zabbix_server]
	filesystem / free percent	OK	10-06-2012 14:35:33	7d 1h 5m 51s	1/4	zabbix agent key 41.590674: vfs.fs.size[/,pfree]
	filesystem /home free percent	OK	10-06-2012 14:35:33	7d 1h 5m 51s	1/4	zabbix agent key 88.759399: vfs.fs.size[/home,pfree]
	zabbix agent	OK	10-06-2012 14:35:30	0d 3h 25m 54s	1/4	TCP OK - 0.046 second response time on port 10050
	zabbix total update percent	OK	10-06-2012 14:34:27	0d 14h 36m 57s	1/4	zabbix agent key 70.89: update_percent

架构的演变

- Master-child to server-proxy
- MySQL to oracle
- Proxy HA
- RPC HA (dns)
- Nagios on Zabbix

Why Server-Proxy

- 最早使用的是master-child，但是有个非常严重的问题。在child上的host会突然消失。经过分析发现可能是与master同步conf信息造成的（not sure）。从之前ppt可以看到master-child的方式是需要master和child同步conf信息的，这个同步是双向的，怀疑是这个出问题。

Master-Child Issue

- 仔细看下文档：
  - Each Child Node periodically sends configuration changes, historical data and events to its Master Node.
  - Each Master Node (a node with at least one child) periodically sends configuration changes to Child Nodes either directly or via other Child Nodes directly connected to the Master Node.
  - Zabbix does not send configuration of a Master Node to Childs.
- 我的理解是，child的conf信息会同步到master，master也会与child同步该child的conf信息。Master本身的conf信息不回同步到child。
- 由于我国奇特的网络状况，怀疑是conf同步出了问题，两边conf不等。从而child认为已经没有了这个host，就将其删除。

Master-Child不足

- Master-child有许多信息分散在不同child的local数据库中，不便数据分析。
- Child也是个Zabbix，太heavy。维护成本高。每台Server重启需要15分钟左右。
- Host消失问题没办法解决。

Use Proxy

- Proxy安装快（cp，sed conf file），启动快（1分钟内），非常轻量。
- Local数据库仅作中转，所有数据汇总到中央数据库，便于数据分析。
- 减小网络问题影响，proxy没有conf信息，全部集中在server。
- Proxy减小数据库io压力。Proxy数据打包发往Server。
- 仅有一个Server，维护方便。

MySQL to Oracle

- 一开始图简单，直接用了MySQL，用了一个月发现撑不住了（vps2000+）。而且，同事中没有MySQL的resource，都是搞Oracle的。比较了下，换了Oracle。之后一直都比较平稳。
- 之后随着Zabbix数据越来越多，有很多数据分析需求，增加了一台Standby的Oracle做数据分析用。主库只用作Server监控。

RPC HA

- Zabbix本身有PHP的API，但用起来很麻烦。Brett Lentz写过最初版本的python api。但因为写的不是很好，所以我们重新了api。https://github.com/baniuyao/ZabbixPythonApi
- Zabbix的API接口是在其frontend上的。我们有两套frontend，通过dns解析做切换。

Performance Tuning

- Zabbix的瓶颈在数据库IO。
  - 提高item的interval
  - 数据库Patch Code (src & php)
  - Configuration parameter
- Apache问题

提高item的interval

- 这个是最简单粗暴、治标不治本，同时也是最快捷的方法。
- 我们现在Zabbix的规模，vps只有1800。数据库还有20%左右的冗余。在初期，我们很多item的interval是60s，数据库撑不住。
- 而且，在实践中发现，就算interval很小，比如10s，也不能做到10s监控一次。
- 一般的item，300s，600s就可以了。安全方面的监控60s。

Our Interval

- 这里看看我们的interval分布情况（半年前的数据）：

- SQL>select delay, count(*), round(count(*)/(select count(*) from items where status=0)*100,2)||'%' as percent from items where status=0 group by delay order by 2 desc;

DELAY	COUNT(*)	PERCENT
600	119489	18.04%
86400	103224	15.59%
300	90168	13.62%
2400	79051	11.94%
7200	62286	9.41%
1200	53981	8.15%
120	37741	5.7%
240	34252	5.17%
3600	25578	3.86%
60	19412	2.93%
51840	15251	2.3%
180	8026	1.21%
172800	4125	.62%
900	3344	.5%
120960	3309	.5%
30	2942	.44%
150	16	0%
6000	5	0%
1800	4	0%

数据库

- TX Lock: 造成数据库不更新数据。
- Max open cursor: 代码问题
- 无法share cursor: Oracle会crash
- History表过大: 影响性能

- 上面三个问题我们都通过修改Zabbix源代码fix。但是是一个case-by-case的solution，并不通用。针对server-proxy架构。TX Lock Patch只针对Oracle。
- 我们的代码已经托管至GitHub：https://github.com/baniuyao/Zabbix_PPTV

TX Lock

- 之前PPT说过，每次trigger判断都会生成一个event，而event需要一个唯一的id。
- Zabbix是怎样做的呢？数据库有一张ids表，记录了当前最大id，在我需要一个eventid时，它的poller（获取数据的进程）会“预支”一定量的id，比如目前最大id是1000，那它会将1000-1255都预留给自己，等自己的event用完后（如果不够，则再重复一遍）。这样，event多以后，多个poller互相竞争，会造成大量的TX锁。
- PS：event的多少与trigger成正比。即规模

Solution

- 使用Oracle的sequence代替这种获取id的方式。
- PHP中也有一些代码需要fix。
- 注意，Master-Child架构可能会出问题，因为每个Child自己的sequence最终可能会有id冲突。

Max Open Cursor

- 这是个很恶心的问题，这问题很简单，就是没关cursor，但解决起来很麻烦，因为不知道到底哪里没关，还要看逻辑。

Solution

- 找到开cursor的地方，然后读代码，最终找到了个地方由于逻辑问题，会跳过close cursor的语句。Fix掉。

大SQL造成无法Share Cursor

- Zabbix对于很长的SQL有个很奇怪的特性，它会将它们拼起来，然后收尾加上begin…end去执行。这样的结果就是oracle无法在share pool找到可以share的cursor。影响性能。

Solution

- 一开始试图在拼SQL的地方让它不要拼接，这个弄了半天没有成功（因为不熟悉C）。后来换了思路，在执行SQL的地方重新将其split掉，分别执行。
- PS：代码中碰到end认为是结束，结果有张表叫“Trends”，中枪了。。。Debug了一天被我火眼金睛看到。

History表过大

- 首先要了解History表和Trends表的关系。History表是存储详细历史的，比如item的interval是600s，那么每600s，就会忘History表插一条记录。Trends是保留数据的趋势，每小时记录一次max，min，avg。
- Zabbix自己有Housekeeper机制，定时会清空History表。但Housekeeper效率非常低，我们在Server上是关闭的（Proxy上是开启的）

Solution

- 我们每周truncate history表。这个truncate的周期可以自行调整。
- 清空History表会造成详细数据缺失。但是从我们使用来看，过去一周的数据不会看的这么详细，1个小时一个点已经足够了。

Configuration Parameter

- 这方面没有什么发言权，因为我们只是调大了参数，并没有做压力测试。而且现在很稳定，没有调整的需要。
- 一台proxy的性能，只要在挂了以后数据可以追的回来就没有问题（追的速度>数据进入的速度）。没有定量分析过。
- 以我的经验，4核/8G虚拟机不要超过800台。
- 可以参考这篇文章：http://blog.zabbix.com/monitoring-how-busy-zabbix-processes-are/457/

Apache问题

- 官方文档是使用Apache的。但在使用过程中，发现apache性能很差，后来我们迁移到了nging+php-fpm，性能好了很多。最起码不会挂起僵死。但php-fpm的high load还是靠crontab的restart解决的。
- Web界面做大的更新后端依然后有TX锁。这个没有找到解决办法，只能等。

SYSTEM TO PLATFORM

Zabbix的角色

- 不可否认，Zabbix是个非常优秀的监控系统，但它想把所有的事情都做好，这就使得它不能focus在最基本的监控上。从跨入2.0的release notes也可以看出，Zabbix对large scalability的问题没有多大兴趣，他们希望做中小型公司的整体运维解决方案。
- 在PPTV，Zabbix逐渐演变为一个监控数据收集、展示和远程crontab的工具。很多功能被弱化（Screen、报警等）。通过api和其它系统集成使得Zabbix更强大。

Zabbix Python API

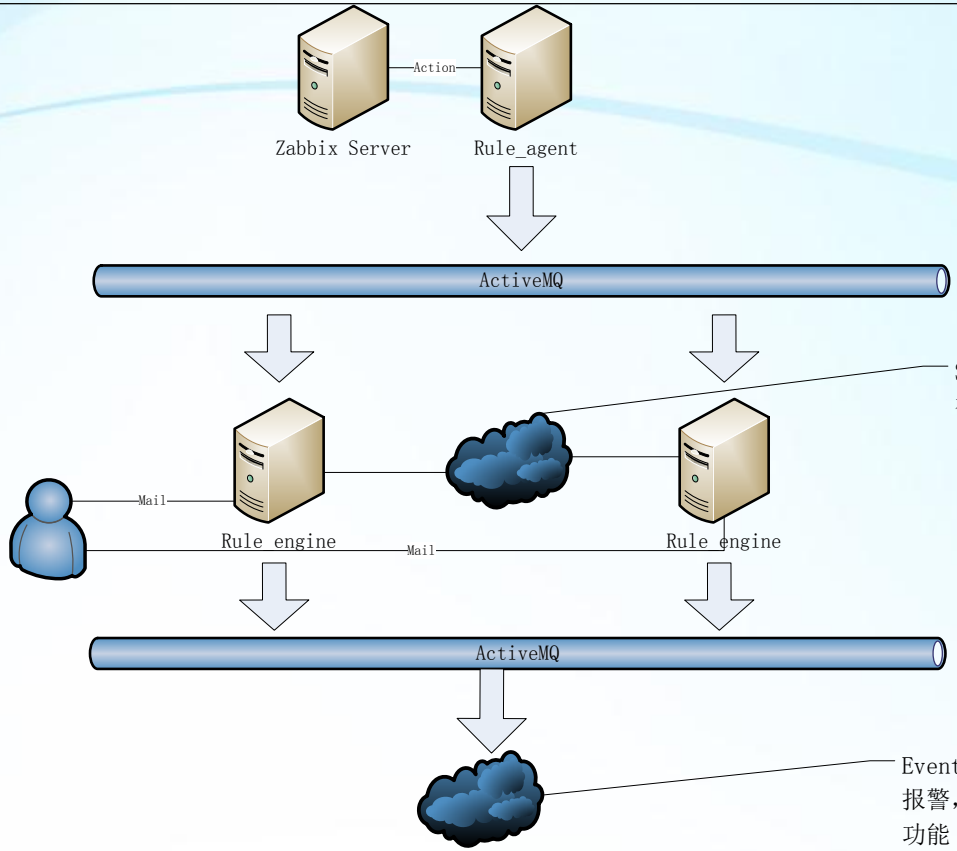
- 我们开发都使用Python，所以在Zabbix的API基础上开发了Python的API。
- https://github.com/baniuyao/ZabbixPythonApi

报警系统

- Zabbix本身有邮件报警系统，但是功能有限。我们是这样做的，action调用脚本，再做处理。
- 对于报警，我们有rule_engine来处理报警，针对不同的报警，会生成不同的邮件、报警等级，负责人，其他信息等。
- 之后报警会发送到event console，集中处理报警。
- 各个系统之间都使用ActiveMQ做消息传输。

报警系统架构

2012年10月5日星期五

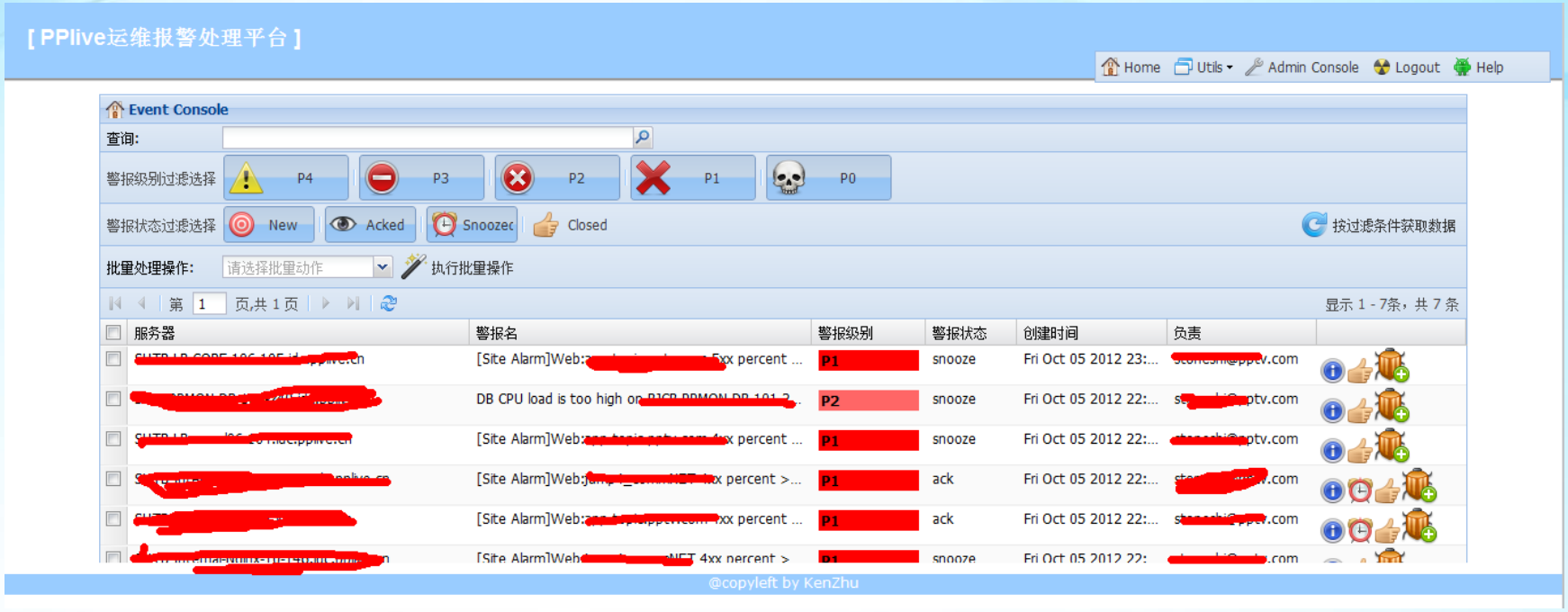


Snooze_console：报警被snooze后，在指定时间内将不再发送报警

Event_console用来展示报警，ack，auto_fix等功能

Event Console

- Ken大神@不装A不装C 搞的。不只Zabbix，其他很多系统都使用它。



Rule_engine

- 报警系统的核心是rule_engine。它的作用是根据不同的规则来定义报警的内容。下图是代码片段，可以看到定义规则的方法。

```
RE = RuleEngine()
ping_check_rule = Rule(parseTriggerName,isDualCheckProblem,bool_func_args={"trigger_name":"unreachable"})
p0_trigger_severity_rule = Rule(parseTriggerName, setSeverity, bool_func_args={"trigger_name":["p0test"]}, action_func_args={"severity":"0"})
p1_host_severity_rule = Rule(parseHostName, setSeverity, bool_func_args={"host_name":["lb","drag","jump", "vod-memcache"]}, action_func_args={"severity":"1"})
p1_trigger_severity_rule = Rule(parseTriggerName, setSeverity, bool_func_args={"trigger_name":[".*IP forward check failed.*","mysql.*availability","redis:.* is no
affic", "dns", "site alarm"]}, action_func_args={"severity":"1"})
p3_host_severity_rule = Rule(parseHostName, setSeverity, bool_func_args={"host_name":["vod-sn", "web-cdn", "live2"]}, action_func_args={"severity":"3"})
p4_trigger_severity_rule = Rule(parseTriggerName, setSeverity, bool_func_args={"trigger_name":["puppet"]}, action_func_args={"severity":"4"})
RE.register(ping_check_rule)
RE.register(p1_host_severity_rule)
RE.register(p1_trigger_severity_rule)
RE.register(p3_host_severity_rule)
RE.register(p4_trigger_severity_rule)
RE.register(p0_trigger_severity_rule)
RED = RuleEngineDaemon('/tmp/rule_engined.pid',RE.filter,stdout='/tmp/rule_engine_stdout.log', stderr='/tmp/rule_error.log')
```

优点

- 不同报警不同处理。Syslog报警需要知道syslog的内容。Lb 4XX太高需要分析日志，cpu load报警要看cpu top等等。Rule_engine提供了框架来处理这些逻辑的问题。
- 使用了ActiveMQ，启动多个instance即可增加吞吐量。

Rule_engine问题

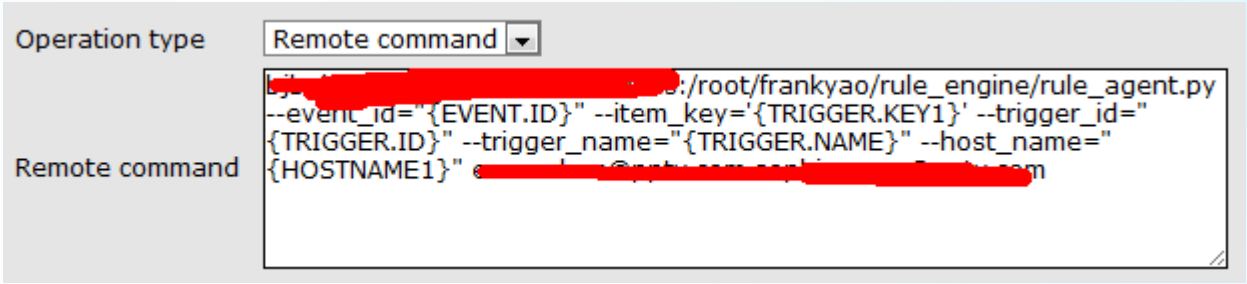
- 代码是用Python去年写的，现在看有很多问题。但由于这个系统很脆弱，不敢轻举妄动。
- Unreachable: Unreachable是指一台机器down了之后的报警。由于网络波动，95%是误报。我们的策略是一旦出现这种报警，rule_engine会进行多点检查，确认是否机器down了。
- 网络出现波动时，大量unreachable报警，直接将运行rule_engine的机器冲垮。就算有amq做pending，但依然撑不住。

改进

- 两个月前写了一套erlang+RabbitMQ的rule_engine，没有经过测试，没有付诸使用。
- Python处理各种复杂的逻辑有点吃力，看过prolog来实现，增强rule_engine的功能（想法没有实践）。
- 高可用太过简单（多个instance），系统不健壮。
- ActiveMQ迁移到RabbitMQ？两者没有详细考察过，只是都看到说rmq好。

如何报警

- Zabbix本身的action支持直接发送邮件，但邮件内容非常简陋。我们使用的方法是将报警的内容作为参数传给一个脚本，通过脚本来进行后续的报警工作。



报警邮件

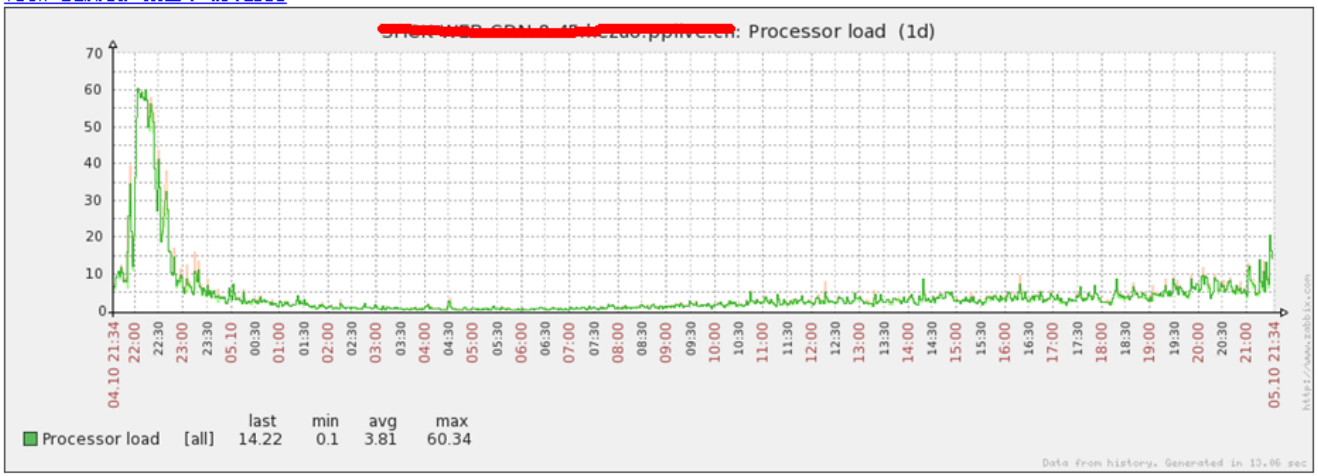
Host: S[redacted]

IP: [redacted]

Issue: CPU load is too high on S[redacted]

Date: Fri Oct 5 21:34:11 2012

View zabbix chart details



Comment:

CPU Load(2 min avg) > 3 * CPU Count
View Data in Application: CPU-Process Load

CPU Status(1 min avg):

报警

- 报警邮件中还有cpu top, mem top, io top等等信息。
- 除了邮件，报警还会在event console中显示。Event console可以ack报警，auto-fix命令（比如php问题可以点下按钮直接restart）。

FUTURE

整合之路

- 继续同其他系统集成。关闭修改Zabbix的对人接口（doing now），keep Zabbix data clean and pure。
- 流程化，自动化，服务化。
- Monitor as Service，Alert as Service（done with ActiveMQ）。

报警系统优化

- Python来写的不稳定。现在是靠crontab来做keep live。对ActiveMQ不了解，RabbitMQ？考虑中。
- 代码的refactor。
- Prolog？Erlang？Robust？

Proxy HA丢数据

- 当一台proxy出现故障暂停服务一段时间后。将这台proxy监控的机器全部迁移到另一台proxy后，这些机器在proxy down的时间内的数据将全部丢失。
- Proxy出现问题时，一般都是优先恢复服务，老数据先不管，但Zabbix的策略是先恢复老数据，再追数据。这就有可能出现数据永远追不上的情况。
- 直接update数据库来更换proxy会使得Server crush。

API性能

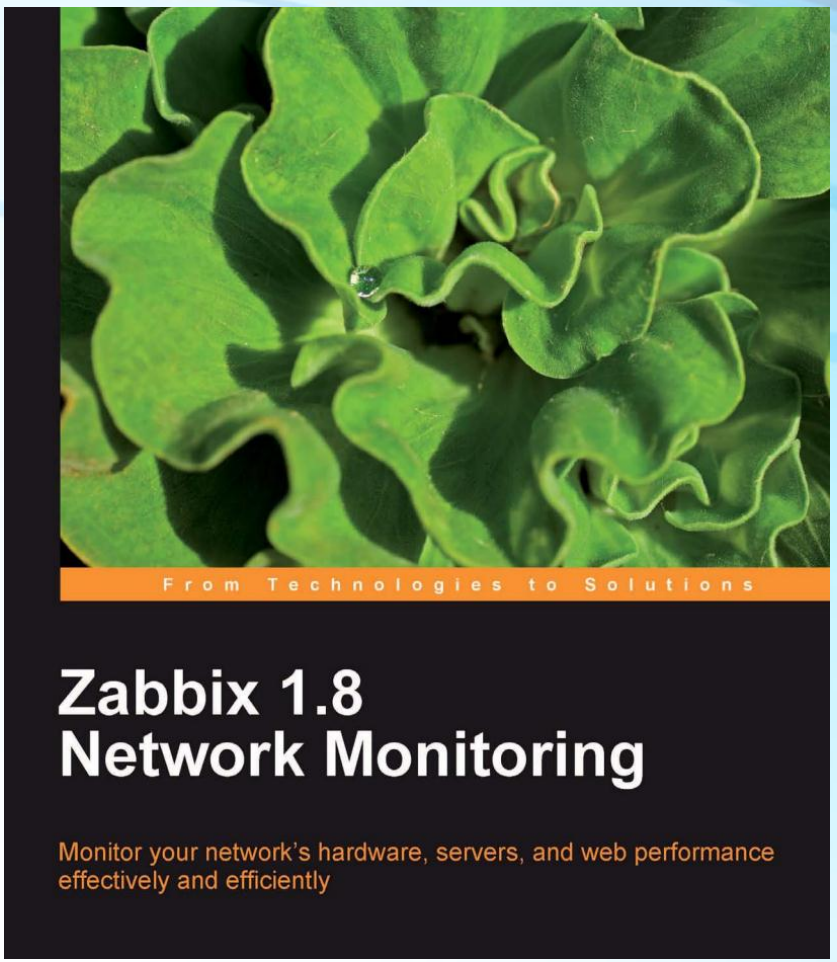
- Zabbix API是基于PHP的，所以一旦高并发，PHP会成为瓶颈。这个我想过基于数据库重新写一套API，但Zabbix会将一些configuration信息读取到内存，直接update数据库会造成两边信息不对等而crush。
- 这个暂时只能通过thread pool解决，效率不高。

Frontend性能

- 大批量更新很容易失败，由于会产生TX锁，需要等待非常长的时间。有时候等了很久却失败。这时需要停掉Server，再在页面上处理。或者直接update数据库。
- 这是由于Zabbix的SQL普遍写的很烂造成的。

About Zabbix

- 相比其他开源产品，Zabbix的社区不够活跃，问题得不到很好的解决。
- 相关资料较少，只有一本书可以参考。



3年后怎么办

- 目前Oracle冗余还有20%左右。3年后这个架构怎么支撑？Oracle的IO如何处理？MySQL集群？

SUMMARY

Summary

- Zabbix对于中小规模的运维是个很好的解决方案，能够很快速的build。
- 资料文档少，要自己摸索。
- Zabbix有许多坑，要亲身经历过才知道。
- 一定要对Zabbix的数据库表结构有清晰的理解。
- 有API可以方便的二次开发。

Thank you!

@超大杯摩卡星冰乐
baniu.yao@gmail.com
2012.10.6