
# 运维超级管理工具Super Agent

窦喆 搜狐网络运营部
Weibo.com @南非蜘蛛

# Agenda

- 开发目的
- 运维管理的挑战
- Puppet简介
- Sagent简介
- Sagent逻辑图
- Sagent注册模块
- Sagent认证模块
- Sagent管理模块
- Sagent插件模块
- Sagent监控模块
- Sagent配置管理模块
- Sagent报警模块
- Slog模块
- Sagent安全性
- 演示
- Q&A

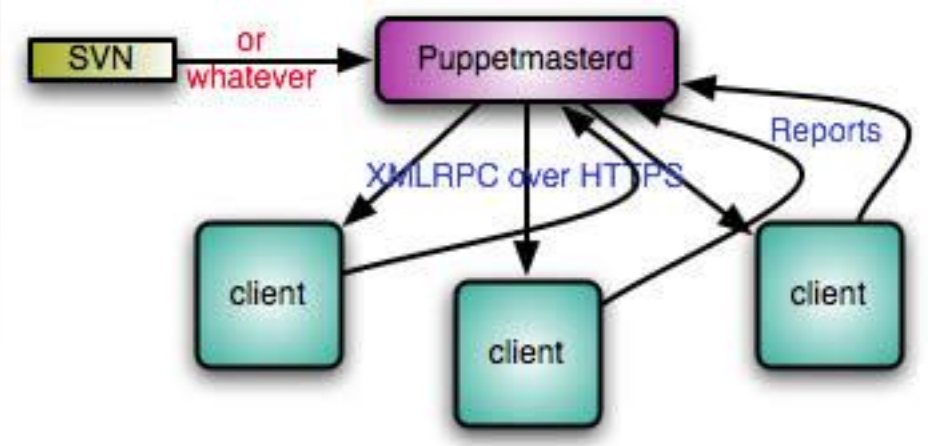
# 开发目的

- 简化运维管理
- 科学的运维管理
- 运维是一种管理艺术

# 运维管理（技术层）的挑战

- 管理员少
- 硬件设备数量多
- 硬件设备类型多
- 硬件设备品牌多
- 操作系统多
- 软件多
- 批量执行
- 实时监控平台
- 实时报警平台
- 配置的集中管理
- 海量的日志
- 现有的运维工具不能满足所有需求

# Puppet简介




- 采用ruby编写，GPL授权
- c/s架构
- 跨平台，Resource Providers
- 配置管理中心
- 扩展能力强
- 多样化解释型语言结构

```
file { '/etc/hosts':
    owner  => 'root',
    group  => 'root',
    mode   => 644,
}
```

# Puppet资源类型

1. augeas
2. computer
3. cron
4. exec
5. file
6. filebucket
7. group
8. host
9. interface
10. k5login
11. macauthorization
12. mailalias
13. maillist
14. mcx
15. mount
16. nagios_command
17. nagios_contact
18. nagios_contactgroup
19. nagios_host
20. nagios_hostdependency
21. nagios_hostescalation
22. nagios_hostextinfo
23. nagios_hostgroup
24. nagios_service
25. nagios_servicedependency
26. nagios_serviceescalation
27. nagios_serviceextinfo
28. nagios_servicegroup
29. nagios_timeperiod
30. notify
31. package
32. resources
33. router
34. schedule
35. selboolean
36. selmodule
37. service
38. ssh_authorized_key
39. sshkey
40. stage
41. tidy
42. user
43. vcsrepo
44. vlan
45. yumrepo
46. zfs
47. zone
48. zpool

参考：**http://docs.puppetlabs.com/references/stable/type.html**

# Puppet存在的问题

- Puppet使用ruby开发，解析型语言，速度慢
- 开发人员不熟悉ruby
- Puppet资源定义过于复杂
- 证书认证麻烦
- 不能实时监控agent状态
- 没有接口和公司其他系统做交互
- 偏重配置管理，不适合发布指令性的命令，比如编译软件，给文件改名，Capistrano 更适合
- 不能远程直接执行非交互式命令

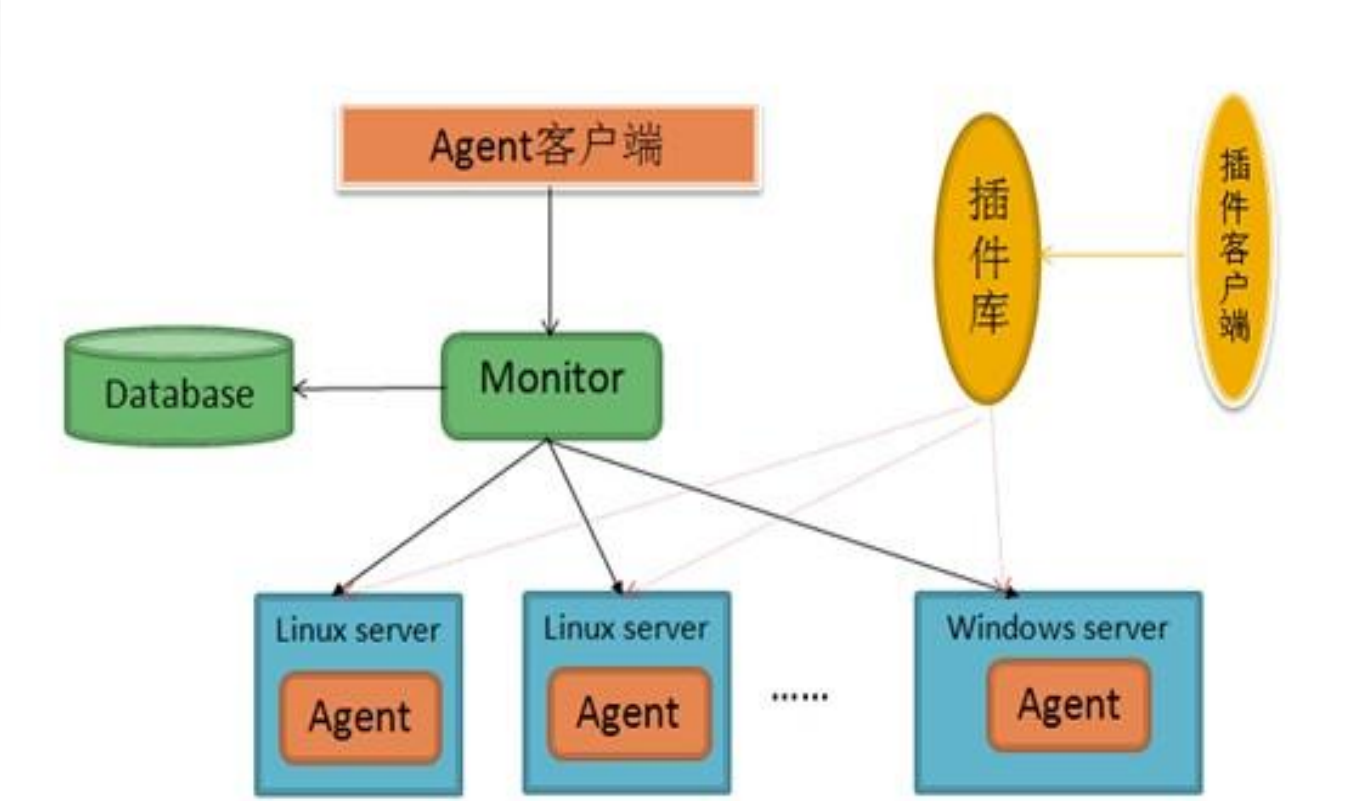
# Sagent简介

- 运行方式C/S
- 支持Linux和Windows平台
- 用C语言开发，效率高

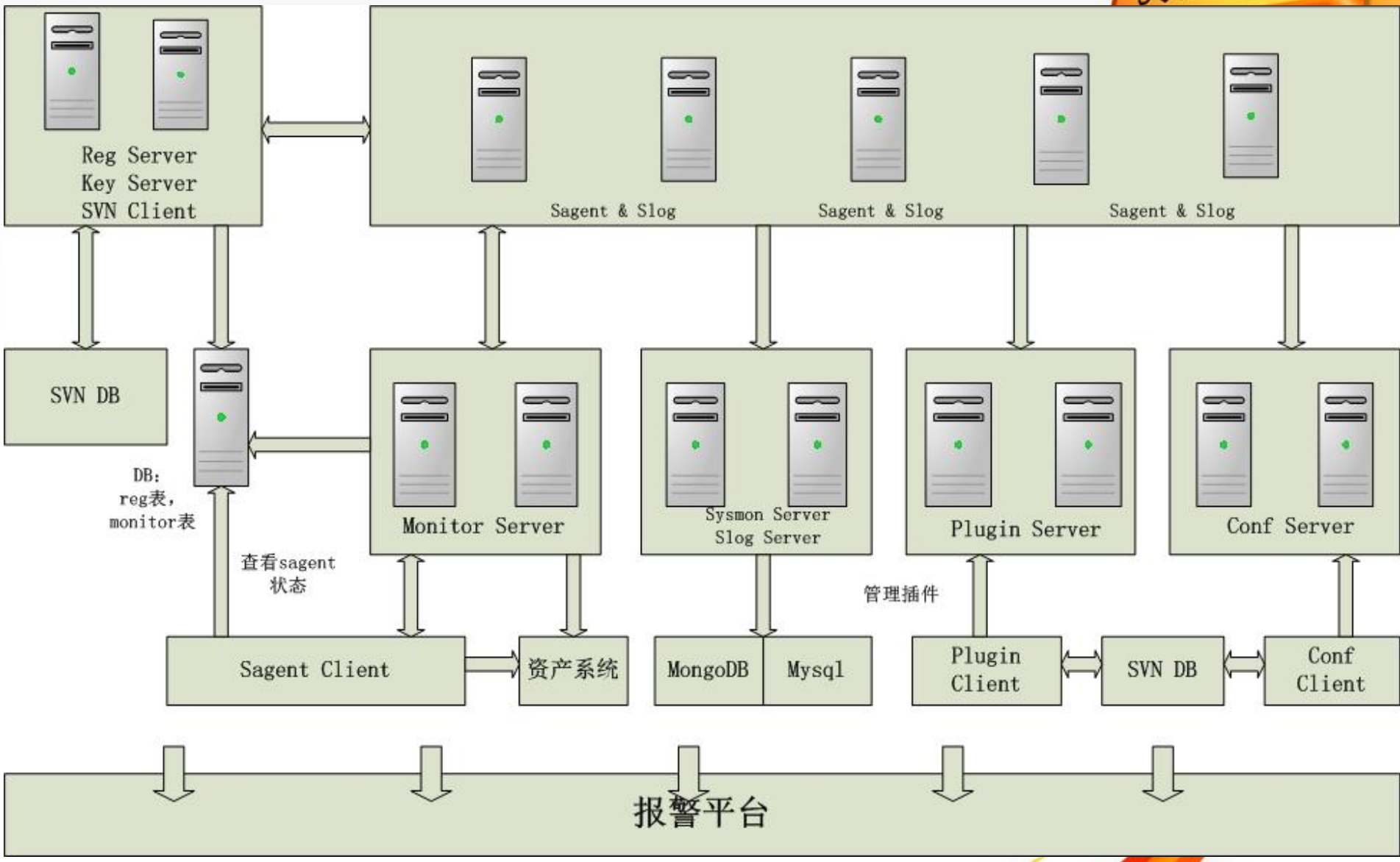
可用性、可靠性、可扩展性、安全性

- 给资产、监控、日志系统等提供数据
- 批量执行命令
- 支持主动和被动方式采集数据
- 实时监控Sagent状态
- 功能服务都支持多台
- 整合配置管理CM
- 横向扩展能力Scale out
- 断网本地保存数据，网络恢复后发送数据
- 安全性高：自主开发，传输加密，权限，认证，ip限制
- Key Server集中认证，Key Server cache设计

# Sagent第一版逻辑架构图



# Sagent第二版逻辑架构图



# Sagent注册模块

- 安装sagentd
- 激活sagentd
- 启动sagentd
- 停止sagentd
- 重启sagentd
- 卸载sagentd
- 重置连接KEY
- 相关文件和功能

# Sagentd启动、停止和重启服务

- sagentd  启动服务
- sagentd -s 停止服务
- sagentd -r 重启服务

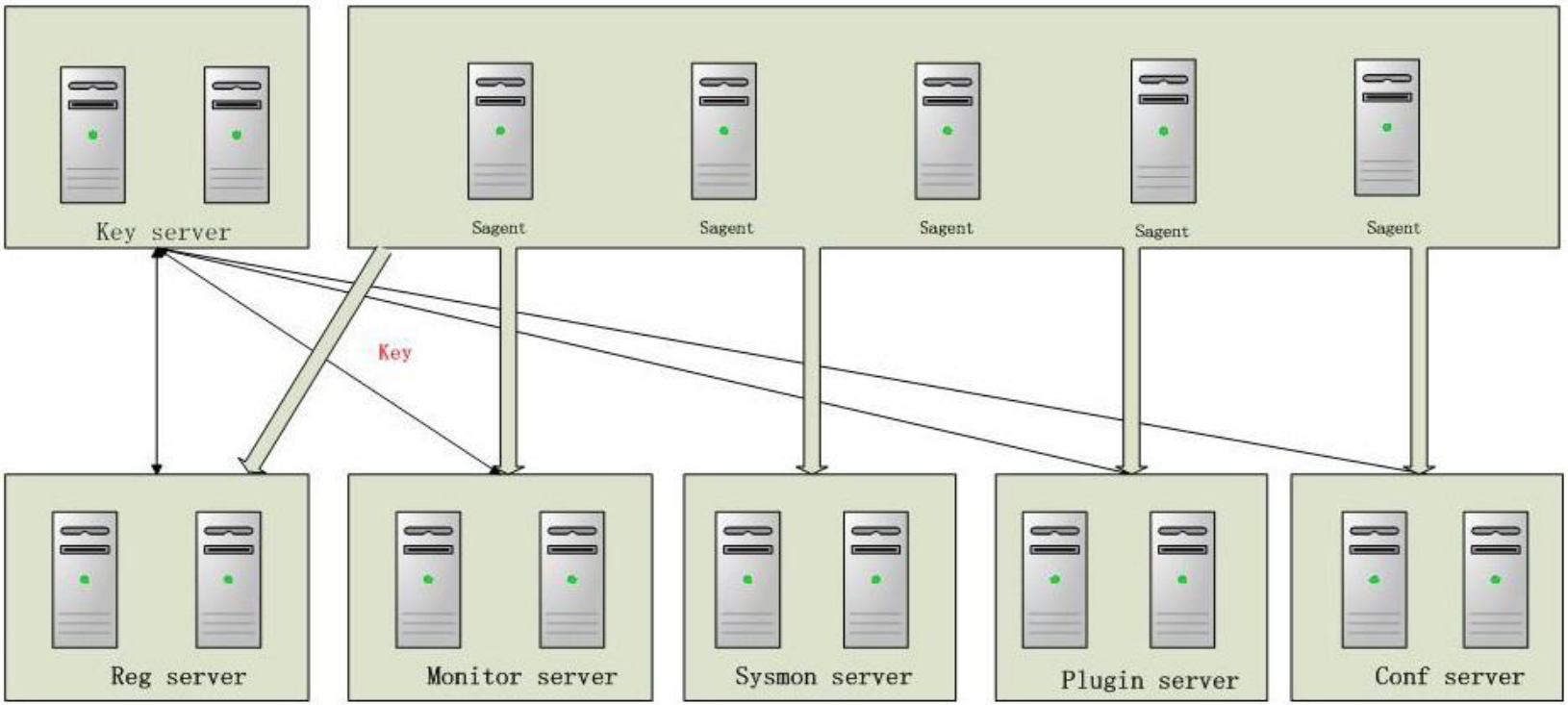
# Sagentd卸载

- SagentServer上运行sagentd_tool -u，到RegServer申请删除，本地删除sagend相关配置文件并停止服务
- RegServer运行sreg –u可查看等待删除
- 运行sreg -D sagend_id 在reg和monitor表里同时删除相关信息

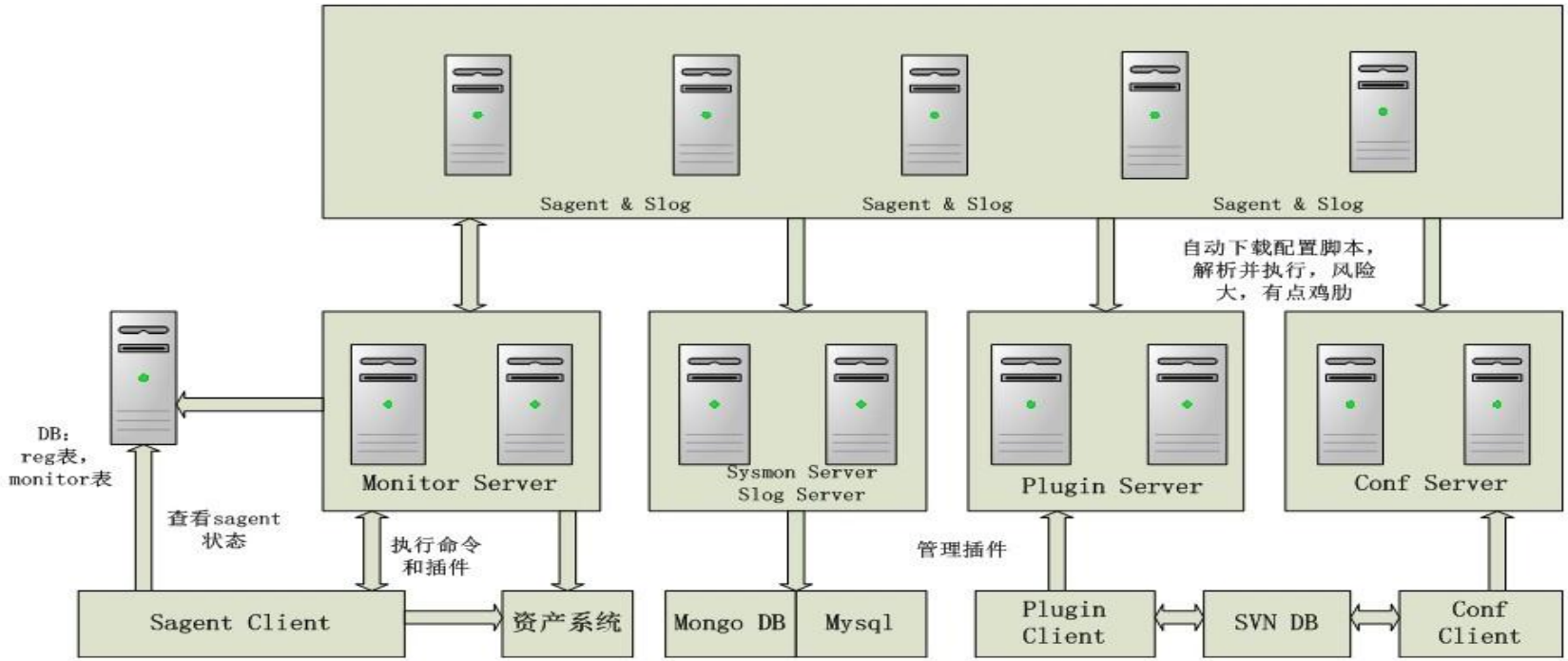
# Sagent Server相关文件和功能

- sagentd_tool -i 安装sagentd
- sagentd_tool -u 卸载sagentd
- sagentd_tool -r 重置连接Key
- sagentd 运行服务
- sagentd -s 停止服务
  sagentd -r 重启服务
- sagentd_sregd_key 安装时是安装Key，认证时是连接Key
- sagentd.conf sagentd从Reg Server得到的相关配置
- policy.conf sagentd从Reg Server得到的模板文件
- config 存放Reg Server的IP
- controller.conf sagentd可以自己定义访问控制权限,默认没有这个文件，但是如果有，就会读取并应用

# Sagent认证模块


Key server
Sagent
Sagent
Sagent
Sagent
Sagent
Key
Reg server
Monitor server
Sysmon server
Plugin server
Conf server

# Sagent管理模块


Sagent & Slog
Sagent & Slog
Sagent & Slog
自动下载配置脚本，
解析并执行，风险
大，有点鸡肋
DB：
reg表，
monitor表
Monitor Server
Sysmon Server
Slog Server
Plugin Server
Conf Server
查看sagent
状态
执行命令
和插件
管理插件
Sagent Client
资产系统
Mongo DB
Mysql
Plugin
Client
SVN DB
Conf
Client

# Sagent插件模块

- Plugin Client向Plugin Server提交插件
- Plugin Server用Key和IP列表控制Plugin Client访问
- Plugin Server为插件生成md5，sagentd执行时会校验
- 提交插件同时提交到SVN Server
- Sagentd定时从Plugin Server同步所有插件
- Sagentd Client控制Sagentd更新指定插件，相当于推
- 插件没分目录，都扔在根目录，有点乱

# Sagent监控模块

- 可以设置为主动和被动方式
- 主动方式：Sagentd定时运行插件获取监控对象和值，值和阀值对比，如果超过阀值，就给报警数据库和监控数据库各发一份，如果没有超过阀值，只给监控数据库发
- 被动方式：Sagent Client去抓取，然后发送给Sysmon Server
- Sysmond可实时入数据库，也可当buffer size满时一次写入数据库
- 使用MongoDB作为数据库，分为报警数据库和监控数据库

# Sagent监控模块

- Sagentd每次在查询一个服务的状态时，产生一个子进程，并且它使用来自该命令的输出和执行程序返回码来确定具体的状态。状态码的含义如下所示：
  0— 表示服务正常地工作。
  1—底，表示服务处于低风险。
  2—高，表示服务处于高风险。
  3—unknown，表示不可用的命令参数或者插件内部运行错误
- 输出用|分隔，比如： 0|httpd status:ok|check_http|2000，支持多行，Nagios 3开始也支持多行
- Sagentd根据输出的状态码决定是否入报警库

# Sagent监控模块

- Policy.conf配置文件：

* * * * * 插件路径/插件 参数1 参数2
* * * * * 插件路径/插件 参数1 参数1

-c代表critical –w代表warning

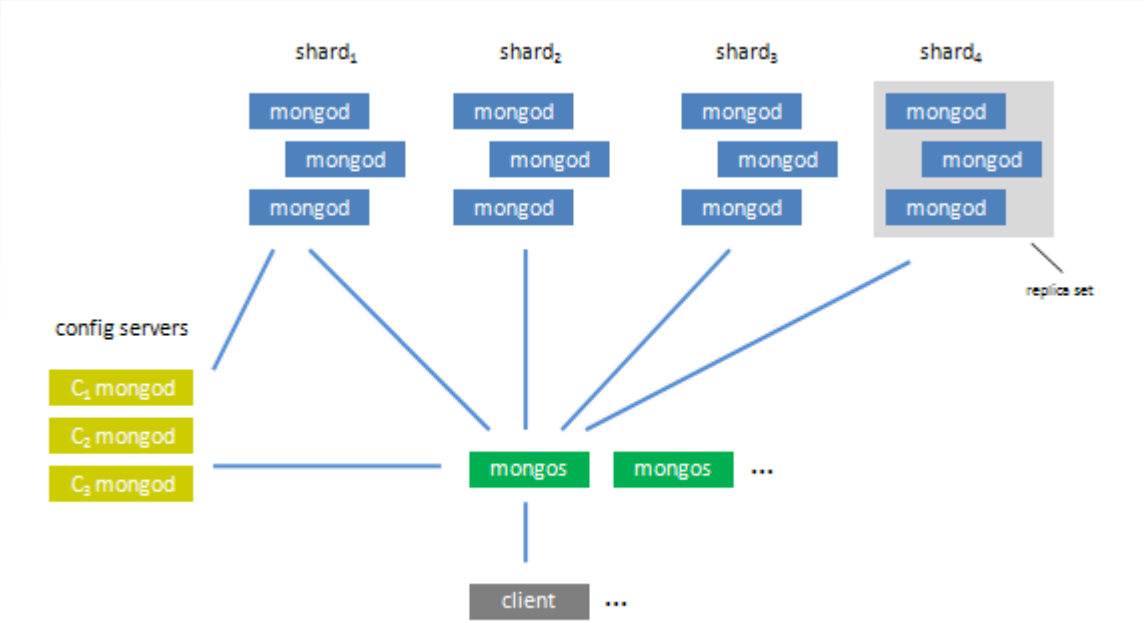
| Range definition | Generate an alert if x... |
|---|---|
| 10 | < 0 or > 10, (outside the range of {0 .. 10}) |
| 10: | < 10, (outside {10 .. ∞}) |
| ~:10 | > 10, (outside the range of {-∞ .. 10}) |
| 10:20 | < 10 or > 20, (outside the range of {10 .. 20}) |
| @10:20 | ≥ 10 and ≤ 20, (inside the range of {10 .. 20}) |

# Sagent监控模块

| Command line examples | Meaning |
| --- | --- |
| check_stuff -w10 -c20 | Critical if "stuff" is over 20, else warn if over 10 (will be critical if "stuff" is less than 0) |
| check_stuff -w~:10 -c~:20 | Same as above. Negative "stuff" is OK |
| check_stuff -w10: -c20 | Critical if "stuff" is over 20, else warn if "stuff" is below 10 (will be critical if "stuff" is less than 0) |
| check_stuff -c1: | Critical if "stuff" is less than 1 |
| check_stuff -w~:0 -c10 | Critical if "stuff" is above 10; Warn if "stuff" is above zero |
| check_stuff -c5:6 | The only noncritical range is 5:6 |
| check_stuff -c10:20 | Critical if "stuff" is 10 to 20 |

**Critical级别高，一旦满足就停止对Warning值的比较**

# MongoDB架构图




1. 一个或多个分片，其中每个分片持有部分数据（自动管理）。读写操作自动路由到合适的分片上。每个分片是一个replica set。
一个replica set是一台或多台服务器，每台机器持有相同数据的拷贝。在特定的时间点，一台机器是主节点而其他机器是从节点。如果主节点死掉了，其中一台从节点自动接管为主节点。所有的写操作和一致性读操作都进入主节点，而所有的最终一致性读操作分布到所有从节点上。
2. 多台配置服务器，其中每台配置服务器持有表明数据位于哪个分片的元数据的拷贝。
3. 一个或多个路由器，其中每个路由器都作为一个或多个客户端的服务器。客户端向路由器发起查询和更新，路由器询问配置服务器后将请求分发到合适的分片上。
4. 一个或多个客户端，其中每个客户端都是用户应用程序的一部分，它使用自身语言的mongo客户端驱动向路由器发起请求。

**mongod 是服务器端程序（数据或配置）。 mongos 是路由器程序。**

# Sagent配置管理模块

- 遵循“promise”方式的变更管理。
- 被管理机器上的配置状态，都是由管理员“预定义”好的策略
- 用SVN类似软件做版本控制
- 简化puppet资源定义

# Sagent报警模块

- 接收监控平台报警信息
- 接收Slog平台报警信息
- 接收Sagent平台报警信息
- 可接受其他平台报警信息
- 报警信息合并，避免报警风暴
- 报警分类：高和底，归并时间不同
- 报警方式：Email、SMS，IM

# Slog模块

**暂时没有开发，考虑先用开源系统，重点考察Cloudera的Flume**

Flume是cloudera于2009年7月开源的日志系统。它内置的各种组件非常齐全，用户几乎不必进行任何额外开发即可使用。

# Flume设计目标

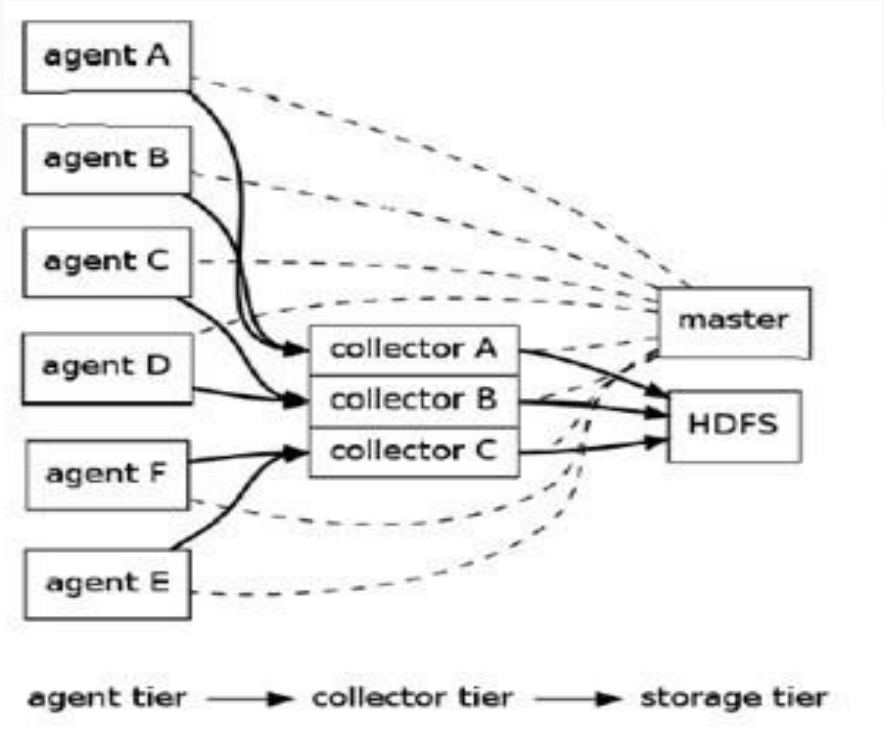
(1) 可靠性
当节点出现故障时，日志能够被传送到其他节点上而不会丢失。Flume提供了三种级别的可靠性保障，从强到弱依次分别为：end-to-end（收 到数据agent首先将event写到磁盘上，当数据传送成功后，再删除；如果数据发送失败，可以重新发送。），Store on failure（这也是scribe采用的策略，当数据接收方crash时，将数据写到本地，待恢复后，继续发送），Best effort（数据发送到接收方后，不会进行确认）。

(2) 可扩展性
Flume采用了三层架构，分别问agent，collector和storage，每一层均可以水平扩展。其中，所有agent和 collector由master统一管理，这使得系统容易监控和维护，且master允许有多个（使用ZooKeeper进行管理和负载均衡），这就避 免了单点故障问题。

(3) 可管理性
所有agent和colletor由master统一管理，这使得系统便于维护。用户可以在master上查看各个数据源或者数据流执行情况，且可 以对各个数据源配置和动态加载。Flume提供了web 和shell script command两种形式对数据流进行管理。

(4) 功能可扩展性
用户可以根据需要添加自己的agent，colletor或者storage。此外，Flume自带了很多组件，包括各种agent（file， syslog等），collector和storage（file，HDFS等）。

# Flume架构




Flume采用了分层架构，由三层组成，分别为agent，collector和storage。其中，agent和collector均由两部分组成：source和sink，source是数据来源，sink是数据去向。

# Flume功能模块

(1) agent
agent的作用是将数据源的数据发送给collector，Flume自带了很多直接可用的数据源（source），如：
text(“filename”)：将文件filename作为数据源，按行发送
tail(“filename”)：探测filename新产生的数据，按行发送出去
fsyslogTcp(5140)：监听TCP的5140端口，并且接收到的数据发送出去

同时提供了很多sink，如：
console[(“format”)] ：直接将将数据显示在控制台
text(“txtfile”)：将数据写到文件txtfile中
dfs(“dfsfile”)：将数据写到HDFS上的dfsfile文件中
syslogTcp(“host”,port)：将数据通过TCP传递给host节点

(2) collector
collector的作用是将多个agent的数据汇总后，加载到storage中。它的source和sink与agent类似。
此外，使用autoE2EChain，当某个collector 出现故障时，Flume会自动探测一个可用collector，并将数据定向到这个新的可用collector上。

(3) storage
storage是存储系统，可以是一个普通file，也可以是HDFS，HIVE，HBase等。

# 开源日志系统比较

| | scribe | Chukwa | Kafka | Flume |
|---|---|---|---|---|
| 公司 | facebook | apache/yahoo | LinkedIn | Cloudera |
| 开源时间 | 2008 年 10 月 | 2009 年 11 月 | 2010 年 12 月 | 2009 年 7 月 |
| 实现语言 | C/C++ | JAVA | SCALA | JAVA |
| 框架 | push/push | push/push | push/pull | push/push |
| 容错性 | collector 和 store 之间有容错机制，而 agent 和 collector 之间的容错需用户自己实现 | Agent 定期记录已送给 collector 的数据偏移量，一旦出现故障后，可根据偏移量继续发送数据。 | Agent 可用通过 collector 自动识别机制获取可用 collector。store 自己保存已经获取数据的偏移量，一旦 collector 出现故障，可根据偏移量继续获取数据。 | Agent 和 collector，collector 和 store 之间均有容错机制，且提供了三种级别的可靠性保证。 |
| 负载均衡 | 无 | 无 | 使用 zookeeper | 使用 zookeeper |
| 可扩展性 | 好 | 好 | 好 | 好 |
| agent | Thrift client，需自己实现 | 自带一些 agent，如获取 hadoop logs 的 agent | 用户需根据 Kafka 提供的 low-level 和 high-level API 自己实现。 | 提供了各种非常丰富的 agent |
| collector | 实际上是一个 thrift server | -- | 使用了 sendfile，zero-copy 等技术提高性能 | 系统提供了很多 collector，直接可以使用。 |
| store | 直接支持 HDFS | 直接支持 HDFS | 直接支持 HDFS | 直接支持 HDFS |
| 总体评价 | 设计简单，易于使用，但容错和负载均衡方面不够好，且资料较少。 | 属于 hadoop 系列产品，直接支持 Hadoop，目前版本升级比较快，但还有待完善。 | 设计架构（push/pull）非常巧妙，适合异构集群，但产品较新，其稳定性有待验证。 | 非常优秀 |

# 开源日志参考资料

scribe主页：https://github.com/facebook/scribe
chukwa主页：http://incubator.apache.org/chukwa/
kafka主页：http://sna-projects.com/kafka/
Flume主页：https://github.com/cloudera/flume/

# Sagent安全性

- 数据传送加密
- IP访问限制
- 认证走不同的密钥
- Failover机制
- 命令权限控制
- Agent Server没有监听端口
- 配置文件使用版本控制

**问题：启动用户是root**

# 演示

安装

```
[@zjm_20_44 ~]# ./sagentd -i
Install successfully! New private key has been saved in /etc/sagent/sagentd_sregd_key
```

等待激活

```
[@zjm_20_44 ~]# ./sagentd
Agent not activated. Please wait...
```

激活

```
[@jn_10_96 ~]# ./sreg
Agent has been activated and template has been applied.
```

# 演示


```
[@zw_72_192 sohu_agent]# ./stat -a
```

| ip | mac_address | ip_from | status | version | monitor_ip | monitor_id | system | arch | update_time | port |
|---|---|---|---|---|---|---|---|---|---|---|
| [illegible] | 00:1F:29:CA:47:E6 | [illegible] | 1 | 1.3 | [illegible] | lt | Linux | x86_64 | 2011-05-23 13:41:10 | 49704 |
| [illegible] | 00:23:7D:A7:DA:2E | [illegible] | 1 | 1.3 | [illegible] | lt | Linux | i686 | 2011-05-23 13:41:10 | 63244 |
| [illegible] | 00:19:B9:EF:F6:00 | [illegible] | 1 | 1.3 | [illegible] | lt | Linux | i686 | 2011-05-23 17:16:58 | 45793 |
| [illegible] | 00:11:0A:30:44:D5 | [illegible] | 0 | 1.3 | [illegible] | dx | Linux | i686 | 2011-06-20 17:47:34 | 15601 |
| [illegible] | 00:11:0A:E9:8A:D3 | [illegible] | 1 | 1.3 | [illegible] | lt | Linux | i686 | 2011-05-23 13:41:11 | 5867 |
| [illegible] | 00:1F:29:CC:42:64 | [illegible] | 1 | 1.3 | [illegible] | lt | Linux | x86_64 | 2011-05-25 15:30:33 | 24547 |
| [illegible] | 00:1F:29:C5:D7:28 | [illegible] | 1 | 1.3 | [illegible] | lt | Linux | x86_64 | 2011-05-25 15:34:46 | 31782 |
| [illegible] | 00:1E:C9:CD:12:6F | [illegible] | 1 | 1.3 | [illegible] | lt | Linux | x86_64 | 2011-06-10 10:16:59 | 8014 |
| [illegible] | 00:26:B9:29:2E:91 | [illegible] | 0 | 1.3 | [illegible] | other | Linux | x86_64 | 2011-06-01 11:29:53 | 63961 |
| [illegible] | A4:BA:DB:2F:12:13 | [illegible] | 1 | 1.3 | [illegible] | dx | Linux | x86_64 | 2011-06-03 14:42:16 | 8080 |
| [illegible] | 00:26:B9:29:2E:8F | [illegible] | 1 | 1.3 | [illegible] | lt | Linux | x86_64 | 2011-06-21 10:44:00 | 30107 |
| [illegible] | 00:17:A4:3A:BC:CC | [illegible] | 1 | 1.3 | [illegible] | dx | Linux | i686 | 2011-05-23 13:41:18 | 33501 |
| [illegible] | 00:16:35:5B:F3:D6 | [illegible] | 1 | 1.3 | [illegible] | dx | Linux | i686 | 2011-05-25 15:32:45 | 64900 |
| [illegible] | 00:1C:C4:45:F1:CE | [illegible] | 1 | 1.3 | [illegible] | dx | Linux | x86_64 | 2011-05-23 13:41:18 | 29522 |
| [illegible] | 00:1B:78:CB:51:AC | [illegible] | 1 | 1.3 | [illegible] | dx | Linux | i686 | 2011-05-31 00:38:38 | 41397 |
| [illegible] | 00:1F:29:69:B8:EE | [illegible] | 1 | 1.3 | [illegible] | dx | Linux | x86_64 | 2011-05-25 15:31:42 | 56522 |
| [illegible] | 00:1E:4F:44:65:09 | [illegible] | 1 | 1.3 | [illegible] | dx | Linux | x86_64 | 2011-05-25 15:35:52 | 63021 |
| [illegible] | 00:1F:29:C4:F3:52 | [illegible] | 1 | 1.3 | [illegible] | dx | Linux | i686 | 2011-05-25 15:33:36 | 35614 |
| [illegible] | 00:19:BB:27:9E:D2 | [illegible] | 1 | 1.3 | [illegible] | dx | Linux | i686 | 2011-05-23 13:41:18 | 32121 |
| [illegible] | 00:26:B9:29:2D:B2 | [illegible] | 1 | 1.3 | [illegible] | dx | Linux | x86_64 | 2011-05-23 13:41:18 | 42641 |
| [illegible] | CA:96:A2:BA:C8:46 | [illegible] | 1 | 1.3 | [illegible] | dx | Linux | x86_64 | 2011-05-23 13:41:18 | 9737 |
| [illegible] | 00:16:36:1C:D9:FF | [illegible] | 1 | 1.3 | [illegible] | dx | Linux | x86_64 | 2011-05-23 17:12:20 | 6653 |
| [illegible] | 00:0C:29:C3:07:B4 | [illegible] | 1 | 1.5 | [illegible] | lt | Windows | x86 | 2011-06-15 10:23:40 | 56941 |
| [illegible] | 00:13:21:B1:B2:D0 | [illegible] | 1 | 1.5 | [illegible] | lt | Windows | x86 | 2011-06-16 16:04:51 | 34983 |
| [illegible] | 00:0F:20:6C:9E:A1 | [illegible] | 1 | 1.5 | [illegible] | lt | Windows | x86 | 2011-06-15 10:23:44 | 26241 |
| [illegible] | 00:16:35:02:3C:02 | [illegible] | 1 | 1.5 | [illegible] | lt | Windows | x86 | 2011-06-20 10:00:46 | 44684 |
| [illegible] | 00:0B:CD:D3:CB:FF | [illegible] | 1 | 1.5 | [illegible] | lt | Windows | x86 | 2011-06-15 10:25:21 | 9113 |
| [illegible] | 00:13:21:CB:A3:9C | [illegible] | 1 | 1.5 | [illegible] | lt | Windows | x86 | 2011-06-17 09:49:57 | 49613 |
| [illegible] | 00:15:60:A5:B7:66 | [illegible] | 1 | 1.5 | [illegible] | lt | Windows | x86 | 2011-06-17 10:09:59 | 61980 |

# 演示

Sagent client 命令

```
[@zw_72_192 sohu_agent]# ./sagent
Usage:
./sagent -f iplist_file | -m ip_addr [-F ip_from] -c command | -p plugin [-a file1,file2,...] [-d outputdir] [-P prefix] [-t threads] [-T timeout] [-N]
```

Splugin client 命令

```
[@zw_72_192 sohu_agent]# ./splugin
Usage:
./splugin [-D dir] [-d] [-y] plugin
./splugin [-l]
```


# Q & A
# THINKS!

**EMAIL：zhedou@sohu.com**
**微博：http://weibo.com/spider4k**