Optimizing your Monitoring and Trending tools for the Cloud

Nagios World Conference 2012

Nicolas Brousse
Lead Operations Engineer
September 28th 2012
• About TubeMogul
• What are some of our challenges?
• Our environment
• Amazon Cloud Environment
• Automated Monitoring
• Efficient on-call rotation
• Efficient monitoring
• What’s next?
• Q&A
About TubeMogul

• Founded in 2006
• Formerly a video distribution and analytics platform
• TubeMogul is a Brand-Focused Video Marketing Company
  – Build for Branding
  – Integrate real-time media buying, ad serving, targeting, optimization and brand measurement

TubeMogul simplifies the delivery of video ads and maximizes the impact of every dollar spent by brand marketers

http://www.tubemogul.com
What are some of our challenges?

• Monitoring between 700 to 1000 servers
• Servers spread across 6 different locations
  – 4 Amazon EC2 Regions (our public cloud provider)
  – 1 Hosted (Liquidweb) & 1 VPS (Linode)
• Little monitoring resources
  – Collecting over 115,000 metrics
  – Monitoring over 20,000 services with Nagios
• Multiple billions of HTTP requests a day
  – Most of it must be served in less than 100ms
  – Lost of traffic could mean lost of business opportunity
  – Or worst, over-spending...
Our environment

- Over 80 different server profiles
- Our stack:
  - Java (Embedded Jetty, Tomcat)
  - PHP, RoR
  - Hadoop: HDFS, M/R, Hbase, Hive
  - Couchbase
  - MySQL
- Monitoring: Nagios, NSCA
- Graphing: Ganglia, sFlow, Graphite
- Configuration Management: Puppet
Amazon Cloud Environment

- We use EC2, SDB, SQS, EMR, S3, etc.
- We don’t use ELB
- We heavily use EC2 Tags

```
ec2-describe-instances -F tag:hostname=dev-build01
```
Automated Monitoring

Configuring Ganglia using Puppet templates

globals {
  ...
  override_hostname = <%= scope.lookupvar('hostname') %>
  ...
}

udp_send_channel {
  host = <%= scope.lookupvar('ec2_tag_nagios_host') %>
  port = 8649
  ttl = 1
}

sflow {
  udp_port = 6343
  accept_jvm_metrics = yes
  multiple_jvm_instances = yes
}
Automated Monitoring

Or configuring Host sFlow using Puppet templates

```plaintext
sflow{
    DNSSD = off
    polling = 20
    sampling = 512
    collector{
        ip = <%= ec2_tag_nagios_host %>
        udpport = 6343
    }
}
```
Automated Monitoring

- Puppet configure our monitoring instances
  - We use Nagios regex: `use_regexp_matching=1`
  - But we don’t use true regex: `use_true_regexp_matching=0`
  - We use NSCA with Upstart

```bash
# Nagios NSCA
description "Nagios NSCA Daemon"
start on network
stop on runlevel [!2345]
respawn
respawn limit 10 5
exec /opt/nagios/bin/nsca -c /opt/nagios/etc/nsca.cfg --daemon
```

- We don’t use the perfdata
- We use pre-cached objects
- We includes our configurations from 3 directories
  - objects => templates, contacts, commands, event_handlers
  - servers => contain a configuration file for each server
  - clusters => contain a configuration file for each cluster

```bash
# OBJECT CONFIGURATION FILE(S)
cfg_dir=/opt/nagios/etc/objects
cfg_dir=/opt/nagios/etc/servers
cfg_dir=/opt/nagios/etc/clusters
```
Automated Monitoring

Process of event when starting a new host and add it to our monitoring:

1. We start a new instance using Cerveza and Cloud-init
2. Puppet configure Gmond or Host sFlow on the instance
3. Our monitoring server running Gmond and Gmetad get data from the new instance
4. A Nagios check run every minute and check for new hosts
   - Look for new hosts using EC2 API
   - Look for EC2 tag “hostname” to confirm it's a legit host, not a zombie / fail start
   - Look for EC2 tag “nagios_host” to see if the host belong to this monitoring instance
5. If a new host is found:
   - We build a config for the host based on a template file and doing some string replace
   - Once all config have been generated, we rebuild pre-cache objects and reload Nagios
6. If we find “Zombie” host, we generate a Warning alert
7. If the config is corrupt, we send a Critical alert
Automated Monitoring

```
# # HOST DEFINITION - Config file managed by check_tm_cluster.py script - DO NOT CHANGE MANUALLY!!
# # Define a host for the local machine

define host{
    use                  linux-server
    host_name            #HOSTNAME#
    hostgroups           #CLUSTERNAME#
    alias                #FQDN#
    address              #IP#
    _DNSVAL              #IP#
    display_name         #CLUSTERNAME# #HOSTNAME#
    _PAGING              yes
    notes                #HOSTNAME# is part of #CLUSTERNAME#. Health check using SSH.
    _AWSID               #AWSID#
}
```
Automated Monitoring

```plaintext
# Define an optional hostgroup for Linux machines

define hostgroup{
    hostgroup_name  mysql-% ec2_placement_availability_zone %>cluster
    alias          MySQL <% ec2_placement_availability_zone.upcase %> Cluster
}

define hostextinfo{
    host_name  ^mysql[0-9]+
    notes_url  /ganglia/?c= <% ec2_placement_availability_zone %>&h=$HOSTALIAS$
}

define service{
    use          passive-service
    host_name    ^mysql[0-9]+  
    service_description  disk_mysql
    display_name  Disk space on /mysql
    servicegroups  services-status
    is_volatile  0
    flap_detection_enabled  0
    max_check_attempts  1
    notifications_enabled  0
}
```
Efficient on-call rotation

• Follow the sun
  – Some of our team is in Ukraine, no more Tier 1 night on-call for us

• Nagios timeperiod and escalation are a pain to maintain
  – Nagios notification plugged to Google Calendar
    • Using our own notification script for email and paging
    • Google Calendar make it easy for each team to manage their own on-call calendar
    • Support for multiple Tier and complex schedules
    • Caching Google Calendar info locally every hour
  – Simpler definitions and rules in Nagios contacts
  – Notify only people on-call, unless they asked for “off call” emails
Efficient on-call rotation

Using Google Calendar...
Efficient on-call rotation

• Simple contact definitions
• Google Calendar info
• Tier Filter (Regex)
• Tier Interval (time to wait before escalating alert since last tier)
• Off call email
Efficient on-call rotation

Who is on-call right now?

If multiple Tier of same level, pick one. If a Tier doesn't answer, escalate to next Tier. DON'T GIVE UP!!!

ops

On Call Tier 1: SR
On Call Tier 2: JF
On Call Tier 3: NB

stats

On Call Tier 1: Mike
On Call Tier 2: Chris

rtb

On Call Tier 1: Nate
On Call Tier 2: Yen
Efficient on-call rotation

On-call contact fetched from Google Calendar at the bottom of the alert makes our life easier!

***** Nagios *****

Notification Type: PROBLEM

Zone: linode
Service: load-average
Host: [redacted]
State: CRITICAL
Address: [redacted]
Duration: 0d 0h 5m 14s
Info: CRITICAL - load average: 27.87, 19.61, 11.81

Date/Time: Thu Sept 27 18:08:25 UTC 2012

Additional notes:
  Load Average

URL: [redacted]

On-Call contacts:
  OPS: On Call Tier 1: MT [redacted]; On Call Tier 2: JF [redacted]; On Call Tier 3: NB [redacted]
  STATS: On Call Tier 1: Mike [redacted]; On Call Tier 2: Chris [redacted]
  RTB: On Call Tier 1: Saba [redacted]; On Call Tier 2: Yen [redacted]
Efficient monitoring

We disable most notification and only care of a cluster status
Efficient monitoring

Most of our checks are based on Ganglia RRD files

define service{
    use generic-service
    hostgroup_name ^[a-z0-9_-]+-cluster
    servicegroups system-status
    service_description mnt_disk_used
    check_command check_rrd!$USER3$/$HOSTGROUPALIAS$/$HOSTALIAS$/mnt-disk_used.rrd!80!90
    notifications_enabled 0
}
define service{
    use generic-service
    host_name <%= hostname %>
    servicegroups cluster-service
    service_description Cluster - mnt disk used
    check_command check_cluster_service!^.+!^mnt disk used$
    contact_groups noc
}
Efficient monitoring

It become really easy to monitor any metrics returned by Ganglia
Efficient monitoring

We can check cluster status by hosts/services but also per returned messages!

```
# nagios status file check for hosts
define command{
    command_name   check_cluster_service
    command_line   $USER1$/check_nagios_status --host-regex=$ARG1$ --service-regex=$ARG2$ $ARG3$
}

# nagios status file check for services msg
define command{
    command_name   check_cluster_service_msg
    command_line   $USER1$/check_nagios_status_msg --host-regex=$ARG1$ --service-regex=$ARG2$ --msg-filter=$ARG3$ $ARG4$
}
```
Efficient monitoring

Usage: check_nagios_status [options]

Options:
- `-h, --help`  
  show this help message and exit
- `-v, --verbose`  
  Verbose logging. (default: False)
- `--status-file=STATUS_FILE`  
  Path to the Nagios status file. (default: /opt/nagios/var/status.dat)
- `--host-regex=HOST_REGEX`  
  Regex used to filter host name.
- `--service-regex=SERVICE_REGEX`  
  Regex used to filter service description. (default: none)
- `-w WARNING, --warning=WARNING`  
  Warning threshold in percent. (default: 30)
- `-c CRITICAL, --critical=CRITICAL`  
  Critical threshold in percent. (default: 60)
- `-u UNKNOWN, --unknown=UNKNOWN`  
  Unknown threshold in percent. (default: none)

Usage: check_nagios_status_msg [options]

Options:
- `-h, --help`  
  show this help message and exit
- `-v, --verbose`  
  Verbose logging. (default: False)
- `--status-file=STATUS_FILE`  
  Path to the Nagios status file. (default: /opt/nagios/var/status.dat)
- `--host-regex=HOST_REGEX`  
  Regex used to filter host name.
- `--service-regex=SERVICE_REGEX`  
  Regex used to filter service description. (default: none)
- `-w WARNING, --warning=WARNING`  
  Warning threshold in percent. (default: 30)
- `-c CRITICAL, --critical=CRITICAL`  
  Critical threshold in percent. (default: 60)
- `--msg-filter=MSG_FILTER`  
  Regex used to filter plugin output and mark it as error.
Efficient monitoring

Using Graphite Federated Storage

- One place to see all our metrics from all the world
- No delay due to rsync of RRD files
- Graph close to real time, delay only due to rrdcached flushing interval
What’s next?

Some hot topics...

• Do trending alert with Nagios based on Graphite/Ganglia data
• Better automation for non-cloud servers
• Ensure we can scale our monitoring when using hybrid cloud (Eucalyptus) or multiple public cloud provider
• Get a better centralized view of our different Nagios
All this wouldn’t be possible without a strong System Operation team

Andrey Shestakov
Eamon Bisson Donahue
Justin Francisconci
Marylene Tanfin
Nicolas Brousse
Stan Rudenko
Thank you...

TubeMogul is Hiring!

http://www.tubemogul.com/jobs
jobs@tubemogul.com

Follow us on Twitter

@TubeMogul @orieg