

Introduction



- About the Presentation
 - audience
 - anyone with basic nagios knowledge
 - anyone with basic scripting/coding knowledge
 - what a plugin is
 - how to write one
 - troubleshooting
- About Me
 - work at NAS (NASA Advanced Supercomputing)
 - used Nagios for 5 years
 - started at Nagios 2.10
 - written/maintain 25+ plugins

NASA Advanced Supercomputing



- Pleiades
 - 11,312-node SGI ICE supercluster
 - 184,800 cores
- Endeavour
 - 2 node SGI shared memory system
 - 1,536 cores
- Merope
 - 1,152 node SGI cluster
 - 13,824 cores
- Hyperwall visualization cluster
 - 128-screen LCD wall arranged in 8x16 configuration
 - measures 23-ft. wide by 10-ft. high
 - 2,560 processor cores
- Tape Storage pDMF cluster
 - 4 front ends
 - 47 PB of unique file data stored

Ref: http://www.nas.nasa.gov/hecc/

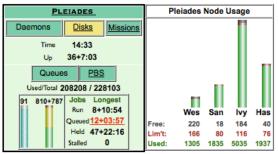
Nagios at NASA Advanced Supercomputing

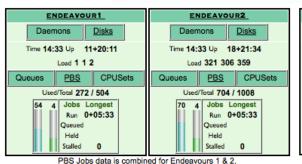


- one main Nagios server
- systems behind firewall send data by nrdp
- some clusters behind firewall
 - one cluster uses nrpe for gathering data
 - other clusters use ssh
- Post processor prepares visualization (HUD) data
 - separate daemon
 - Nagios APIs provide configuration and status data
 - provides file read by HUD
 - general architecture adaptable for other uses

HUD









MEROPE

Up 17+22:41

Used/Total 12096 / 13644

Stalled

10+5 Jobs Longest

Run 0+10:05

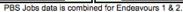
Queued 0+04:36

14:33

Disks

Daemons

Queues





Lou-TB	Lou-MDS1	Lou-MDS2	LFE1	LFE2	LFE3	LFE4	Lou-Mov1	Lou-Mov2	Lou-Mov3	Lou-Mov4	Lou-Mov5	Lou-Mov6	LDAN2	LDAN3	LDAN4	LDAN5	LDAN6	LDAN7	LDAN8	LDAN9	LDAN10
Daemons	Daemons	Daemons	Daemons	Daemons	Daemons	Daemons	Daemons	Daemons	Daemons	Daemons	Daemons	Daemons	Daemons	Daemons	Daemons	Daemons	Daemons	Daemons	Daemons	Daemons	Daemons
Disks	Disks	Disks	Disks	Disks	Disks	Disks	Disks	Disks	Disks	Disks	Disks	Disks	Disks	Disks	Disks	Disks	Disks	Disks	Disks	Disks	Disks



Plugins – Nagios extensions



- Built-in plugins
 - Aren't truly built-in, but they come standard when you install nagios-plugins
 - check_disk
 - check_ping
- Custom plugins
 - Let you test anything
 - The sky's the limit if you can code it, you can test it

What are Plugins?



Nagios configuration to define a service that will use the plugin check_mydaemon.pl:

```
define service {
      host
                          linuxserver2
      service_description
                          Check MyDaemon
      check_command
                          check_mydaemon
define command {
                          check_mydaemon
      command_name
                          check_mydaemon.pl -w 5 -c 10
      command_line
```

Reasons to write your own plugin



- There isn't a plugin out there that tests what you want
- You need to test it differently

Guidelines



- Any Language you want
- There is only one rule: it must return a nagios-accepted value

ok (green)	0
warning (yellow)	1
critical (red)	2
unknown (orange)	3

Plugin Psuedocode



- General outline of what a plugin needs to do:
 - initialize object (if object oriented code)
 - read in the arguments
 - set variables
 - do the test
 - return results
- This is just a suggestion

For Perl: Nagios::Plugin



Instantiate Nagios::Plugin object (the 'usage' parameter is mandatory)

```
my $p = Nagios::Plugin->new(
   usage => "usage_string",
   version => $version_number,
   blurb => 'brief info on plugin',
   extra => 'extended info on plugin'
);
```





```
# adding an argument ex: check_mydaemon.pl -w
# define help string neatly – use below instead of qq
my $hlp_strg = '-w, --warning=INTEGER:INTEGER\n'.
               'If omitted, warning is generated.';
$p->add_arg(
    spec => 'warning|w=s',
    help => $hlp_strg
    required => 1,
    default => 10,
#accessing the argument
$p->opts->warning
```

For Perl: Nagios::Plugin (cont).



```
# finishing the script:
$p->nagios_exit(
    return_code => $p->check_threshold($result),
    message => " info on what $result means"
);

# if you are not using check_threshold use text for return code:
return_code => 'OK|WARNING|CRITICAL|UNKNOWN'
```

For Perl: Nagios::Plugin (cont).



- When you've done your code and have \$result to compare to the thresholds:
 - \$return_code = \$p->check_threshold(\$result)
 - follows nagios convention of min:max
 - check_mydaemon.pl -w 5 will warn on anything > 5
 - check_mydaemon.pl -w :5 will warn on anything > 5
 - check_mydaemon.pl -w 5: will warn on anything < 5
 - check_mydaemon.pl -w 5:7 will warn on anything <5 or >7
 - if you overlap critical and warning, critical has precedent

Overcoming issues



- Test needs elevated privilege
 - nagios can be run as root but is not secure
 - run the test as root via cronjob; write info to a flat file
 - use nagios plugin to read and process the file
- Output of the test was too big
 - the resulting nrdp command hit a kernel limit
 - use ssh to get the output to the main nagios server ex: ssh blah blah
 - use plugin on the main server to process it

Nagios perfdata



- Nagios is designed to allow plugins to return optional performance data in addition to normal status data
 - in nagios.cfg enable the process_performance_data option.
 - Nagios collects this information to be displayed on the GUI
 - in the format "|key1=value1,key2=value2,...,keyN=valueN
 - this can be anything that has a numerical value

Troubleshooting



- The Nagios display says: return code XXX is out of bounds
 - your script returns anything other than 0,1,2,3
 - otherwise it is a nagios error.
 - Google is your friend
 - ex: 13 usually means a permission error
 - sometimes all it tells you is "something went wrong"
 - these disappeared at our site when we switched to Nagios::Plugin
- try running the plugin from the command line
 - verify who you are running as
 - verify the arguments passed in

Troubleshooting (cont).



- Timing is everything!
 - launching too many processes
 - files can get overwritten
 - by cron jobs
 - by multiple nagios processes
- if perfdata is enabled, the perfdata log is the most useful

Questions



Any Questions