

Site Configuration and Anomaly Detection

Matúš Harvan

Networks and Distributed Systems Seminar, 15 May 2006

- 1 cfengine
 - site configuration

- 2 cfenvd
 - anomaly detection

- proliferation of TCP/IP networks
- cheap UNIX-like solutions
- how to keep track of hundreds or thousands of systems and be sure they are configured correctly
- changes/fixes done by hand or via shell scripts – cumbersome and haphazard as number of systems increases

- high level language-based system administration tool
- configuration of all hosts in a central file
- configuration language hides differences between different operating systems
- automates frequently performed system maintenance tasks
- document and enforce the characteristics, interrelationships and dependencies of all hosts in a single, easily readable file
- class-based decision making – optimized for large number of hosts as well as pin-pointed systems

Functionality Summary

- network interface testing and configuration
- text file editing
- symbolic link management
- file permission and ownership testing and setting
- deletion of garbage files
- mounting of NFS filesystems
- other sanity checks

Why a new language?

- provide relevant tools for a limited number of frequently used operations
- frees programs from checking code
- admits a more conceptually user-friendly interface

Symbolic links

- cfengine: `file1 -> file2`
 - Does link exist? If not, create it.
 - Is the name a plain file or directory, not a link? If so signal a warning.
 - If link exists, does it point to the location specified?
 - If yes, do nothing, say nothing.
 - If not, signal a warning.
- shell command: `ln -s file2 file1`

Symbolic links

- cfengine: `file1 -> file2`
 - Does link exist? If not, create it.
 - Is the name a plain file or directory, not a link? If so signal a warning.
 - If link exists, does it point to the location specified?
 - If yes, do nothing, say nothing.
 - If not, signal a warning.
- shell command: `ln -s file2 file1`
- the cfengine version can be run an unlimited number of times without generating spurious and uninteresting output
- the same could be done with a shell script
- required code is simplified
- irrelevant details are hidden

- class-based decision structure
- traditionally a large `if...then...else`
- cfengine runs on each host and determines into which classes it belongs
- statements labeled with classes (no formal decision structures)
- classes
 - hostname
 - operating system and architecture
 - user-defined host groups
 - day of the week
 - logical AND of the above
- classes can be defined or undefined on the command line

- resembles a Makefile, targets replaces by classes
- free format file
- sections
 - each dealing with a particular task (e.g. symbolic links or file editing)
 - each defines actions for various classes

`section:`

`class::`

`statements`

- in contrast to Makefiles, dependencies are not files that have to exist, but host attributes
- not instructions how to build a system, but how the system should look like
- possible to set order in which actions are performed

Overview of functionality

- network
- file editing
- mount model
- files and links
- calling scripts

- ethernet interface and resolver configuration
 - netmask
 - bit-convention for determining broadcast address (ones or zeros)
 - default route
 - system domain name
 - ordered list of nameservers

- BSD/System 5 systems configured primarily by human-readable textfiles
- most config files are line-based text files
- higher level editing functions for transparency rather than flexibility
 - DeleteLinesStarting "text..."
 - DeleteLinesContaining "text..."
 - AppendIfNoSuchLine "text..."
 - PrependIfNoSuchLine "text..."
 - WarnIfNoSuchLine "text..."
 - WarnIfLineMatching "text..."
 - HashCommentLinesMatching "text..."
 - HashCommentLinesStarting "text..."
 - HashCommentLinesContaining "text..."
 - SlashCommentLinesMatching "text..."
 - PercentCommentLinesMatching "text..."
 - ...

File editing example - motd

- prevent changes to the “message of the day” file on a GNU/Linux system
- comment out the respective line in the startup script
editfiles:

```
linux::  
{ /etc/rc.d/rc.S
```

```
HashCommentLinesContaining "motd"  
}
```

- similar to sed or perl, but more high level
- nothing really new introduced (functionality-wise)
- common interface
- using class selectors
- readability

- (NFS) mount procedure automated
- two types of mountable resources
 - **binary filesystems**
 - architecture specific data, e.g. compiled software
 - **home filesystems**
 - user's login areas, meaningful on any type of host
- user-definable pattern to distinguish between binary and home fs
- defined as `server:/site/server/fsname`
- resources mounted on directories with the same names
 - not a restriction (if using a rational naming scheme)
 - anomalies – miscellaneous mount command (awkward syntactically, lifts naming convention)
- adherence to naming convention prevents name clashes
- symbolic links used for cosmetic changes (i.e. alias from `server2:/site/serv2/local` to `/usr/local`)
- editing exports files not addressed by cfengine directly

- files
 - check the existence, ownership and permissions
 - files or directories with controlled recursion
 - fixing or sending warnings to administrator
 - setuid-root and setgid-root files monitored
- links
 - tests, makes and destroys links
 - can link all files in a directory to another directory
- tidy
 - systematic deletion of garbage files (/tmp, /var/tmp)
 - can specify age of files

- possible to write scripts in cfengine language
- example:
 - users managing their own files
 - opening files for collaboration
 - closing private files

- use as many global rules as possible, avoid exceptions
- more difficult and specific tasks better dealt with locally, e.g. a (cfengine) script
- administrators have to discipline themselves to make changes via cfengine rather than directly

- use as many global rules as possible, avoid exceptions
- more difficult and specific tasks better dealt with locally, e.g. a (cfengine) script
- administrators have to discipline themselves to make changes via cfengine rather than directly

Summary

- a language based interface for automating key areas of system administration on a potentially large number of hosts
- configuration of all hosts controlled from a single, central program
- make with shell scripts would do the same job – the gain is no need to write long and complicated scripts employing repetitive checking procedures

- 1 cfengine
 - site configuration

- 2 cfenvd
 - anomaly detection

- detecting anomalies on a single host, using for self-regulation
- off-line time series analyses – consumes storage space and CPU time
- update sampled data iteratively (read database, combine with new value, update database)
- database stores average and variance for a fixed window without having to retain each measurement individually
- convergent geometric series – importance of data degrades over time

Two dimensional time

- $t = nP + \tau$
- topological slices of period P
- parametrized by (τ, n)
- mean and std. dev. are functions of τ
- granularity of 5 minutes found to be sufficient for significant changes in user level behavior
- period is a week
- data stored in fast Berkeley database format

- number of users
- number of privileged processes
- number of non-privileged processes
- amount of free disk
- number of incoming sockets for some well known services
- number of outgoing sockets from the same services

Computing averages with roll-off

- goals:
 - approximate an offline sliding-window time-series analysis over a weekly interval
 - minimal load to the system
 - predictable error on uncertainty margin
- update old estimate of the average \bar{q} with new data point q

$$\bar{q} \rightarrow \bar{q}' = (q|\bar{q})$$

$$(q|\bar{q}) = \frac{wq_1 + \bar{w}\bar{q}}{w + \bar{w}}$$

- repeated iteration leads to a geometric progression in

$$\lambda = \frac{\bar{w}}{w + \bar{w}}$$

- hence on each iteration the importance of previous contribution is decreased by λ

Standard deviation computation

- average standard deviation (error)

$$\langle \sigma(\tau) \rangle \equiv \sqrt{\langle (\delta q(\tau))^2 \rangle_N}$$
$$\delta q(\tau) = q(t) - \langle q(\tau) \rangle_N$$

- $\langle \dots \rangle_N$ is pseudo-fixed-window average

- need to store in database

```
double q_bar[i];  
double delta_q_bar[i];  
double time_index;
```

- comparison computation: $\delta q(t) / \langle \sigma(\tau) \rangle_N$

Characterizing pseudo-periodic functions

- pseudo-periodic function with pseudo-period P

$$\begin{aligned}q(t) &= \sum_{n=0}^{\infty} q(nP + \tau) \quad (0 \leq \tau < P) \\ &= \sum_{n=0}^{\infty} \chi_n(\tau)\end{aligned}$$

- τ is on a circular dimension, so two sliding averages
 - over corresponding times in different periods

$$\langle \chi(\tau) \rangle_T \equiv \frac{1}{T} \sum_{n=l}^{l+T} \chi_n(\tau)$$

- of neighboring times in a single period

$$\langle \chi(n) \rangle_P \equiv \frac{1}{P} \sum_{l=\tau}^{\tau+P} \chi_n(l)$$

Characterizing pseudo-periodic functions

- similarly define standard deviations and their sliding window versions
 - over corresponding times in different periods

$$\sigma_T(\tau) = \sqrt{\frac{1}{T} \sum_{n=l}^{n=l+T} (\chi_n(\tau) - \langle \chi(\tau) \rangle_T)^2}$$

- of neighboring times in a single period

$$\sigma_P(n) = \sqrt{\frac{1}{P} \sum_{l=\tau}^{l=\tau+P} (\chi_n(l) - \langle \chi(l) \rangle_P)^2}$$

- difference between average and current value
 - over corresponding times in different periods

$$\delta_T \chi = \chi - \langle \chi \rangle_T$$

- of neighboring times in a single period

$$\delta_P \chi = \chi - \langle \chi \rangle_P$$

Characterizing pseudo-periodic functions

- normal behavior defined as a two-dimensional criteria

$$\{\delta_T\chi(\tau), \delta_P\chi(n)\} < \{2\sigma_T(\tau), 2\sigma_P(n)\}$$

- geometrical form

$$\Delta(\tau, n) = \sqrt{\frac{\delta_T\chi(\tau)^2}{\sigma_T(\tau)} + \frac{\delta_P\chi(n)^2}{\sigma_P(n)}}$$

- classify into concentric, elliptical regions
 - $< \sqrt{2}$
 - $< 2\sqrt{2}$
 - $< 3\sqrt{2}$

- incorporated into cfengine as cfenvd daemon
- examples in cfengine

```
RootProcs_low_dev2
netbiossn_in_low_dev2
smtp_out_high_anomalous
www_in_high_dev3
```
- possible to react to anomalies, i.e. shut down mail server if outgoing SMTP too high
- low activity could indicate problems elsewhere
- patterns
 - Web-server scan for email addresses, followed by an SMTP peak when spam mail was sent
 - outgoing spam attack – outgoing SMTP and user processes high

- double averaging leads to lower false positives
- problem with periods of low activity
 - afterwards everything is abnormal
 - policy decisions needed to renormalize the threshold
- problem with number of processes
 - many programs start multiple processes (e.g. window manager login)

- two-dimensional time slice approach for anomaly detection
- iterative algorithm, lower resource usage as off-line time series analysis
- implementation in cfengine



M. Burgess

A site configuration engine

Computing Systems, 8(3):309337, 1995.



M. Burgess

Two Dimensional Time-Series for Anomaly Detection and Regulation in Adaptive Systems.

In *Proc. 13th IFIP/IEEE Workshop on Distributed Systems: Operations and Management*, Montreal, October 2002.

Springer LNCS 2506.