SLAC-PUB-12379

March 2007

Cfetool: A General Purpose Tool for Anomaly Detection in Periodic Data

Alf Wachsmann
SLAC, Stanford, CA 94025, USA
Elizabeth Cassell
UC Santa Barbara, Santa Barbara, CA 93106, USA

Cfengine's environment daemon "cfenv" has a limited and fixed set of metrics it measures on a computer. The data is assumed to be periodic in nature and cfenvd reports any data points that fall too far out of the pattern it has learned from past measurements. This is used to detect "anomalies" on computers.
We introduce a new standalone tool, "cfetool", that allows arbitrary periodic data to be stored and evaluated. The user interface is modeled after rrdtool, another widely used tool to store measured data. Because a standalone tool can be used not only for computer related data, we have extended the built-in mathematics to apply to yearly data as well.

1. Introduction

Ever since Mark Burgess introduced an environment daemon to cfengine [1] we felt the need to feed other kind of data into it than the about 20 hard-coded metrics. The hard-coded observables in cfenvd are things like network traffic to and from certain ports, memory and disk usage, number of users on a computers, etc. If one wanted to add other observables, new code needed to be added to cfenvd which makes it very inflexible. The mathematical engine in cfenvd [2,3] is used for detecting anomalies in periodic data. It cannot be applied to arbitrary time series data. Cfenvd defines classes for cfagent depending on the amount of difference of the current data point from the learned average behavior for the given time slot in the period. cfagent can then take the appropriate actions on a computer with such an anomaly. Actions include removing files in case too much disk space is used or killing processes in case too much CPU time is consumed.

Another tool - rrdtool - exists, that allows arbitrary data to be fed into it no matter what the source of this data is. Rrdtool [4] comes with a general purpose user interface and the mathematical engine can be applied to this data. However, rrdtool cannot be used for anomaly detection.

An extension to rrdtool implementing the Holt-Winters aberrant behavior detection [5] is a generalization from cfenvd's periodic data analysis. It accounts not only for the periodic component in the data sets but also for a constant offset and a linear component. In total, 5 different parameters need to be hand-tuned to make the "aberrant behavior" detection with Holt-Winters work. This tuning is different for every dataset and needs to be done at the time the data storage is initialized at the very beginning.

Goal for our new tool "cfetool" was to keep the very easy to use mathematical engine from cfenvd and add a general purpose user interface that is close to rrdtool's. The idea was to make cfetool a drop-in replacement for rrdtool where the periodic nature of the observable is known. Additionally, we wanted cfetool to interact with cfagent in the same way cfenvd does. At the same time, we wanted to extend the mathematical model of cfetool to deal with data that exhibits a yearly period. This allows not only computer data to be stored and analyzed but also data from natural phenomena like whether data with its seasonal pattern.

In the next chapter, we describe the implementation details an how we achieved the above goals. In chapter 3 we will describe several different applications to illustrate the usefulness of cfetool. We conclude with describing some potential further enhancements.

2. Implementation Details

Most of the code for cfetool was directly taken from cfenvd and cfenvgraph. However, some changes were necessary to achieve the goals of a good user interface and for data with different periods.

2.1. Extension to Daily and Yearly Periods

By supporting not only computer related data with cfetool, we have decided to extend the only period cfenvd supports ("weekly") by two more periods: "daily" and "yearly".

Inside of cfenvd and cfetool data points are stored in "buckets" which correspond to a certain time step during the weekly/daily/yearly time period. Let's assume that the update frequency for a daily data set is 10 minutes. The first bucket stores the running average of data points from minutes 0-9, the second bucket data for minutes 10-19 and so on. A total of 24 hours * 60 minutes/hour / 10 minutes = 144 buckets are needed to store data over the entire 24 hour period. Together with some bookkeeping information,

Work supported in part by US Department of Energy contract DE-AC02-76SF00515

the running average of each bucket is eventually stored in a cfenvd/cfetool database file.

The daily periods are implemented by simply exposing the internal resolution cfenvd uses to store weekly data via a user interface and then writing the data in this resolution to the database file. No additional changes to the mathematical engine were necessary.

To allow for yearly periodic data, the mathematical formulas had to be added to cope with this longer time frame. Instead of dividing the total number of minutes per week by the update frequency, we had to use the number of minutes per year. This stretches the buckets linearly from just one week to an entire year.

2.2. Monthly Periods

The question arises, why we did not implement a monthly time period as well. A month is a peculiar time construct: it is purely artificial and has no equivalent in nature (the moon phases are a close but no exact match). The first day of each month is not always the same day of the week. Human activity with a monthly period can mean one of two things: something is happening according to the number of the day of the month (paychecks come every 1st and 15th of the month) or it is happening according to the days of a month (every Monday of a month). With this observation, it is not clear how the storage buckets should be mapped to a month: should the fist bucket represent the first (e.g.) 1 hour interval of the first day of the month or should it be the first 1 hour interval of the first (e.g.) Sunday of the month? An additional difficulty is that months have different lengths. One has 28 and sometimes 29 days, others have 30 or 31 days. The lack of measurements for the last days of some months makes the statistical analysis of cfenvd/cfetool for these days much less significant and thus less trustworthy (see [2] and [3] for how this effects the statistics.)

A similar but much smaller problem occurs with February 29th in the yearly period. We decided to keep a bucket for this day and live with the fact that data in this bucket is updated only 1/4th of the time. According to [2], it takes about 4 complete time periods of data for the averaging algorithm in cfenvd/cfetool to output meaningful data. Consequently it takes 4*4=16 years for data from February 29th to be meaningful.

3. User Interface

In order to make cfetool an easy replacement for rrdtool and because many internal procedure are very similar, we decided to model the user interface of cfetool after rrdtool's. The structure of all commands is:

```
cfetool <action> [parameters]
```

Where `<action>` is one of the following items.

"create" is used to create a new and empty database. It takes the name of the metrics as argument. The period of the data can be determined with one or more flags: `[--daily|-d] [--weekly|-w] [--yearly|-y]`. The update interval in minutes `[--step|-s step]` is another possible parameter. If the create action is used with the `[--file|-f]` flag, it takes a file as input which contains measurements to feed into the cfetool database in batch mode. This can be used to quickly feed existing data into cfetool. It is much faster than individual update actions (see below) because intermediate results are not written back into the database file but kept in memory. Only at the very end the data points are written to the file.

"update" inserts a new data point into the database. It takes the metric name and an optional timestamp as arguments. If the timestamp is omitted, the current time will be used. With the `[--daily|-d] [--weekly|-w] [--yearly|-y]` flags one can indicate which database(s) should be updated.

"check" takes a new value and checks it against the averages currently in the database specified by the metric name. The value is not stored in the database. The output indicates how much higher or lower the new value is compared to the averages in the database in terms of the number of standard deviations. This feature allows "what-if" experiments without influencing the stored data. A typical question that can be answered with this action is: If the entire data distribution had already been learned. What would be the amount of deviation from the average behavior for this particular data point? The data point could be an actual data point which was not flagged as anomaly at the time when it was fed into the database but with the knowledge of the entire distribution (i.e. in hindsight) the assessment might be different.

"dump" dumps the content of an entire cfetool database file in XML format. "import" imports such dump back into cfetool's internal database format.

"info" prints out some basic information about the specified cfetool database.

A separate command "cfetoolgraph" outputs graphs of averages for visual inspection of a cfetool database. The output files are in a format viewable by "gnuplot" or "xgmr" or other graphical plotting program. The inconsistency in the user interface compared to rdtool is inherited from the cfengine distribution which we did not want to change.

4. Return Codes and Outputs

Before exiting, “cfetool update” will print one line to the standard output stream in the following format:

```
yrly=ynum,wkly=wnum,dly=dnum
```

`ynum`, `wnum` and `dnum` will be either the number 0 if the corresponding database was not updated, or a code indicating the state of the given statistic, as compared to an average of equivalent earlier times, as specified below:

code	classifier	meaning
-2	-	no sigma variation
-4	low	within noise threshold, and within
-5	normal	2 standard deviations from
-6	high	expected value
-14	low	microanomaly: within noise
-15	normal	threshold, but 2 or more standard
-16	high	deviations from expected value
-24	low	normal; within 1 standard deviation
-25	normal	from the expected value
-26	high	
-34	low	dev1; more than 1 standard
-35	normal	deviation from the expected
-36	high	value
-44	low	dev2; more than 2 standard
-45	normal	deviations from the expected
-46	high	value
-54	low	anomaly; more than 3 standard
-55	normal	deviations from the expected
-56	high	value

Where “low” indicates that the current value is below both the expected value for the current time position and the global average value. “high” indicates that the current value is above those values. “normal” indicates that the current value is within the range of expected values.

“cfetool update” also exits with a code corresponding to the above table. If more than one database is being updated, the most negative result from the updates is returned and the individual results must be obtained from the standard output stream as described above.

The text representation of these return and output codes correspond directly to classes defined by cfenvd which are then used by cfagent to define classes for the use in cfengine scripts.

A monitoring tool other than cfagent can easily compare these return codes and trigger an alarm in case a certain threshold is exceeded.

5. Simultaneous use with cfenvd

Cfetool can be used to feed its anomaly detection into cfagent just like cfenvd does (use the `[--cfenvd|-c]` flag with the create and update action). The internal interface is a simple ASCII file with class definitions for each metrics.

Here an example for a weather data metrics (see the Application section below for details):

```
kasseltemp_high_dev1
kasseltemp_high_dev2
kasseltemp_high_anomaly
value_kasseltemp=15
average_kasseltemp=3.0
stddev_kasseltemp=2.0
```

The metric name is “kasseltemp” with a current value of “15” while the average value for the current time slot is “3.0” with a deviation of “2.0”. Because the current value is far above average, three additional classes are defined classifying the current value as being more than 1 standard deviation too high and more than 2 standard deviations too high and being an anomaly because the value is too high.

To prevent cfetool and cfenvd writing to this interface file at the same time, we have introduced an advisory lock for the entire file.

6. Applications

Similar to rrdtool, there are no limits to what kind of data can be stored and analyzed with cfetool. Thus, the usability goes well beyond computer monitoring.

6.1. Stand-alone use of cfetool

To demonstrate the versatility of cfetool, we have picked long term weather data as an example. The outside temperature presumably follows a seasonal pattern with a yearly period. Cfetool allows us to test this assumption and show which temperature measurements are outside the average value for a specific day and by how much they are outside the expected range.

The data set we used in this example is from the German Weather Service (Deutscher Wetter Dienst; `http://www.dwd.de/de/FundE/Klima/KLIS/daten/online/nat/ausgabe_tageswerte.htm`) We picked the weather station with German zip code 10438 which is in the city of Kassel. We are using the average temperature 2 meter above ground in degree Celsius (column “TM” in the data set). The data ranges from January 1st, 1991 to February 22nd, 2005, spanning 15 years with one measurement per day.

After slightly converting the date and extracting just the TM column, we get a file with lines of this format:

```
02/20/2005 12:00:00 0.3
```

This can now be fed into cfetool using the batch mode with the command:

```
cfetool create kasseltemp --step 1440 \
--yearly --file cfetool.input
```

Because we have only one measurement per day, we are using 24 hours * 60 minutes/hour = 1440 minutes as the step value. The above command results in a new subdirectory in the current working directory called "kasseltemp" with a file "yearly.db".

The command

```
cfetoolgraph kasseltemp --resolution \
--yearly
```

creates a new file `kasseltemp/yearly-snapshot/graph` with the average for each bucket and the standard deviation of all data points falling in that bucket. The `--resolution` flag is used to obtain a high resolution output which prevents cfetoolgraph from smoothing out the data.

A simple Gnuplot script produces the graph depicted in Fig. 1 out of the `graph` file produced by the command above.

This graph shows that the temperature pattern at this particular German weather station indeed follows a yearly pattern with fairly small deviations over the years.

One question that cannot be answered by cfetool is whether there is a global warming effect that increases the average temperature slowly over the years. The slow increase of the average value would result in slowly growing averages with the overall standard deviations growing only very slightly. The mathematical model implemented in cfenvd/cfetool is designed to analyze the periodicity of data and not the long term trends.

Using the database with the yearly distribution, we can now use the "check" functionality of cfetool to look at some specific data points.

Let's test what a temperature of 35^oC would be on September 8, 2003:

```
cfetool check kasseltemp --yearly \
--time 1063047600 --value 35.0
```

Which outputs

```
yrly=-36,bkt=61;wkly=0,bkt=0;dly=0,bkt=0
```

meaning a return code of "-36" for the yearly data set which translates to a temperature which is higher than 1 standard deviation above the average but below 2 standard deviations. This value is compared against the average from bucket "61" representing the values of all September 8th data. The other return codes and bucket information are all zero because we have created only the yearly database.

6.2. Cfetool and cfagent

Cfetool can be used to interface to cfagent in the same way cfenvd does. This way, cfagent can be used to monitor arbitrary observables. Though restricting it to computer related metrics makes the most sense because that is the area where cfagent is used in.

Here is a setup we are using at SLAC to monitor the temperature of our machine room. Our thermal sensor elements are connected via serial lines to a terminal server software which publishes new measurements every 10 minutes via a broadcasting tool. A client tool can receive these broadcasted values and process them further. In this particular case, the client tool feeds the measurements into cfetool with the command

```
cfetool update X --cfenvd --daily \
--weekly --yearly --value Y
```

where X is replaced with the location of the temperature sensor ("underfloor" in the example below) and Y with the actual measurement. Cfetools defaults to using the current time as timestamp, hence no `--time` argument is necessary. The `--cfenvd` parameter tells cfetool to write the evaluation of the most current measurement into the interface file for cfagent as described above. Note that the temperature data is periodic in a daily, weekly and yearly rhythm (it would be constant in an ideal machine room). With this setup, we are monitoring the temperature at the "underfloor" location with the following snippet of cfengine script:

```
alerts:
  monhost.underfloor_high_dev1::
    "Underfloor temperature
     anomaly high 1 dev at $(env_time)
     current value $(value_underfloor)
     av $(average_underfloor)
     pm $(stddev_underfloor)"
```

6.3. Cfetool in conjunction with other monitoring tools

Cfetool can be used from within Perl or Shell scripts to store and evaluate periodic data. In a Shell script, for instance, one can do the following:

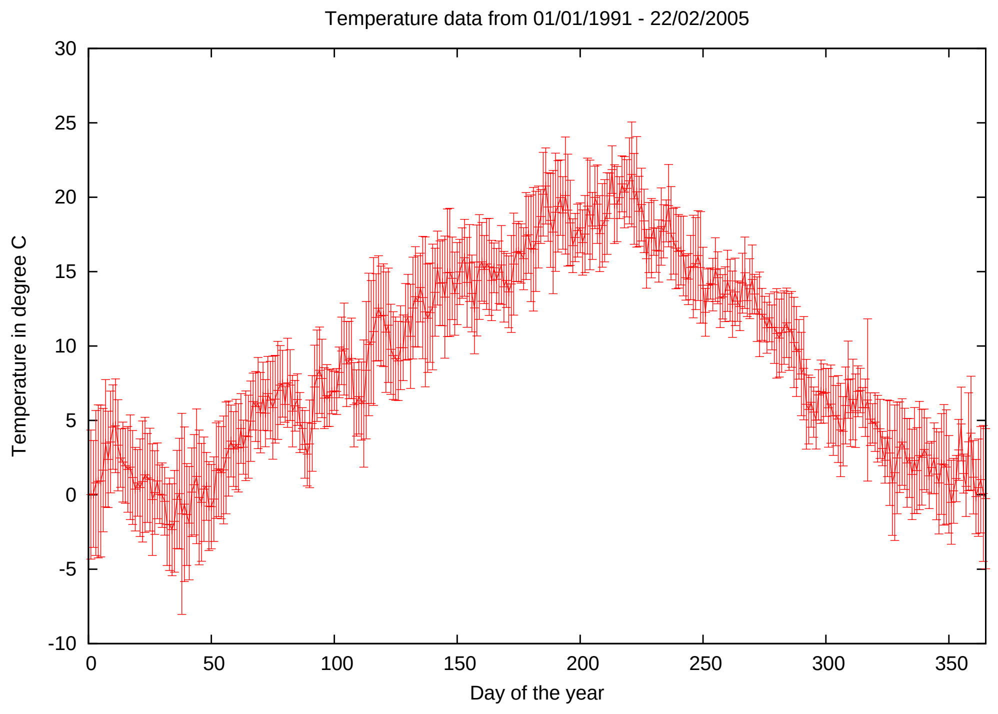

Figure 1: Daily temperature in the German city of Kassel.

```
#!/bin/sh

# acquire the measurement data
XYZdata=‘/some/fancy/tool‘

# store the data with cfetool
# ignore STDOUT but remember the return code
rc=‘cfetool update   \
            --weekly \
            --value $XYZdata >> /dev/null‘

# react in case something is wrong
if [ -30 -gt $rc -a $rc -gt -39 ]; then
  echo "XYZ has value of $XYZdata"   | \
  mail -s "dev1 problem with XYZ" admin
  if [ -40 -gt $rc -a $rc -gt -49 ]; then
    echo "XYZ has value of $XYZdata" | \
    page -s "dev2 problem with XYZ" admin
  fi
fi
```

Because the user interface is very similar to rrdtool, cfetool can also be used as data storage within your favorite host and service monitoring tools like Nagios [6]. In these cases, rrdtool should not be replaced but complemented by cfetool due to its very different scope.

7. Possible Enhancements

It could be desirable to have cfetool bindings to other programming languages like Perl. This could be done by providing a separate library with the necessary functions. With such a library, cfenvd and cfetool could be simplified by removing duplicate source code and linked against the new library.

It would be nice to have a gnuplot template with coloring of data points that lie outside the one and two standard deviation range to make it easier to read the plots from cfenvd and cfetool. The Holt-Winters aberrant behavior detection extension to rrdtool could provide some ideas how to accomplish this.

The inconsistent user interface for cfetoolgraph could be removed. More importantly, this program could be changed to output a graph as graphics in PNG or JPG format directly without the need for gnuplot. This would mirror the behavior of the “rrdtool graph” command.

8. Availability

Cfetool is available as part of the cfengine distribution.

References

[1] M. Burgess. Cfengine Web Site. `http://www.cfengine.org/`.

[2] M. Burgess. Probabilistic anomaly detection in distributed computer networks. Machine Learning Journal (submitted).

[3] M. Burgess. Two dimensional time-series for anomaly detection and regulation in adaptive systems. IFIP/IEEE 13th International Workshop on Distributed Systems: Operations and Management (DSOM 2002)

[4] T. Oetiker. Rrdtool Web Site. `http://people.ee.ethz.ch/~oetiker/webtools/rrdtool/`.

[5] J.D. Brutlag. ”Aberrant behaviour detection in time series for network monitoring. Proceedings of the Fourteenth Systems Administration Conference (LISA XIV) (USENIX Association: Berkeley, CA).

[6] E. Galstad. Nagios Web Site. `http://www.nagios.org/`.