System Evaluation (A2)

System Overview

The system contains two scripts, the anomaly_detection script collects raw *cpu load*, *cpu utilisation*, *memory utilisation* and *network utilisation* data from 17dcompv008 via SNMP for a given period of time. If any measurement is over the specified warning limit then the administrator will be emailed about this abnomality. The statistics script is the same script from coursework part A1 and again, runs various functions offline to calculate statistics on the given data file. Below displays a list showing the script options and internal functions.

- **anomaly_detection.sh [seconds] [max_cpu_load] [max_memory] [max_network]**
  - function *get_cpu_load*
  - function *get_cpu_utilisation*
  - function *get_memory_usage*
  - function *get_bandwidth_in_usage*
  - function *send_email*
- **statistics.sh [file] [cp95] [cp99]**
  - function *read_measurements*
  - function *variance*
  - function *standard_deviation*
  - function *confidence_interval*

Both scripts have been given executable permissions:

```
chmod +x anomaly_detection.sh
chmod +x statistics.sh
```

A **data** folder will be created in the same location as the scripts. This is where collected data files will be stored. The statistics script can then run against these files.

Scripts

anomaly_detection.sh

Similar to part A1, the script first sets the total run duration by getting the system *$SECONDS* property plus the specified user seconds (**$1**). Then, at every *tick* (currently set to 5 seconds), it gets various measurement data: cpu load, cpu utilisation (%), memory utilisation (%) and network utilisation (%). This data is then piped to it's respective named data file.

> **Note:** Each data file is formatted: **[yyyymmdd-hhmmss]-[measureand].txt**, for example *20171111-*

> *213148_cpu_load.txt*.

Three new measurements are being recorded as well as the cpu load from part A1.

CPU Load

See coursework part A1 & **UCD-SNMP-MIB::laLoad.1** MIB resource.

CPU Utilisation

This measurement measures total cpu usage and is tracked via the **HOST-RESOURCES-MIB::hrProcessorLoad** MIB resource; this is the average, over the last minute, of the percentage of time that the processor was not idle.

Memory Utilisation

This measurement measures total memory usage and is tracked via various resources under the **UCD-SNMP-MIB** MIB such as **::memTotalReal**, **::memAvailReal**, **::memBuffer**, **::memCached**. The measurement procedure section below explains how memory usage can be calculated from these four resources.

Network Utilisation

This measurement measures total network usage and is tracked via two resources under the **IF-MIB** MIB, **::ifInOctets** and **::ifSpeed**. The measurement procedure section below explains how network usage can be calculated from these two resources.

The script also takes three extra parameters, max_cpu_load (**$2**), max_memory (**$3**) and max_network (**$4**). If any collected measurement exceeds it's respective limit, an email is sent (using the **sendmail** command) to the admin email address which includes the the abnormal value and time stamp in the email body.

Again, options specified to *snmpwalk* are *-Oqv* so only raw numbers are stored.

For example, to start collecting all measurements over 30 seconds run the anomaly detection script with the following option:

```
./anomaly_detection.sh 30 0.7 15 0.7
```

statistics.sh

The statistics script is unmodified from courseowork part A1 and discussed in the respective document. The script is ran exacly the same way. For example, to run the statistics script on cpu utilisation data use:

```
./statistics.sh /homenfs/pg/b0270122/Postgraduate/data/20171111-213148_cpu_utilisation.txt
1.65 1.96
```

Measurement Procedure & Statistics

Again, the collection of measurements and the statistics are ran independently. CPU utilisation, memory utilisation and network utilisation were chosen as new measurements as they're all very related and may help further explain the servers overall behavior.

The statistics script was ran on each of the four data files after a long period of time. The mean and confidence intervals were then used to *best guess* values to be used for the abnormal limits when starting the script again.

The abnormal limit for cpu load was set to 0.7, this is based from online suggestions which may enable us to proactively fix the issue before becoming overloaded.

Reference:http://blog.scoutapp.com/articles/2009/07/31/understanding-load-averages

The following resources from **UCD-SNMP-MIB** were used to calculate the memory usage: **::memTotalReal**, **::memAvailReal**, **::memBuffer** and **::memCached**.

To calculate actual memory usage the following formula was used:

$$memTotalReal - memAvailReal = totalRAMUsed$$

where:

$$totalRAMUsed - memCached - memBuffer = actualUsedMemory$$

Reference:https://support.solarwinds.com/Success_Center/Network_Performance_Monitor_(NPM)/NET-SNMP_memory_calculation

To get the correct network to poll, a request to **IF-MIB::ifName** was used which returned a list with two items (see below screenshot). The **lo** network is just a loopback device, like a virtual network. The **eno1** is the actual onboard ethernet adapter.

To calculate actual network usage the following formula was used:

$$Utilisation = \frac{\Delta ifInOctets * 8 * 100}{(numberOfSecondsIn\Delta) * ifSpeed}$$

The script keeps track of the elapsed poll time between requests to help with this.

Reference:https://www.cisco.com/c/en/us/support/docs/ip/simple-network-management-protocol-snmp/8141-calculate-bandwidth-snmp.html

Again, the script naively sleeps for 5 seconds on every iteration and therefore does not take in to account the time it takes to do the SNMP request.

> **Note:** The precision of variables within the statistics script are always at their maximum precision. The precision is rounded to two decimal places for console output only.

How this could be extended

The above script is ok when dealing with a single server however, in a multi-server environment with many users (such as a University), many improvements could be made. Firstly, the number of users on the system could also be recorded to view peak times during the day / specified time period. Also, during times of abnormal utilisation, active user ids could also be logged to map suspicious behavior to the anomaly (assuming user ids are allowed to be stored from a data protection point of view).

The statistics gathering is very much offline in the prototype, it would be better if the statistics script was constantly running against the data files while the anomaly detection was also running. This would mean we'd have accurate / up to date statistics and would improve the abnormality values used for detection, as these could then be dynamic.

A real enterprise system would also want to poll from multiple endpoints at once (be this servers, printers or door sensors). Therefore, a better interface for viewing the data would need to be implemented as the console output is fairly basic and has limited use. It would also be good to have a subscription system for the emailing as certain managers may only be concerned with specific abnormalities (a system administrator may not want door detection email warnings).

Lastly, various machine learning techniques exist which could be used against the collected data to give us a better indication of what an abnormality actually is rather than naively stating that if a measurement exceeds a specific value then it is abnormal.

Data & Charts

Sample Output

cpu_load.txt / cpu_utilisation.txt / memory_usage.txt / bandwidth_in_usage.txt

Each measurement will be piped to it's specific output file containing only raw values, one value per line. See the (**Data Files**) section under screenshots to view examples of all four files.

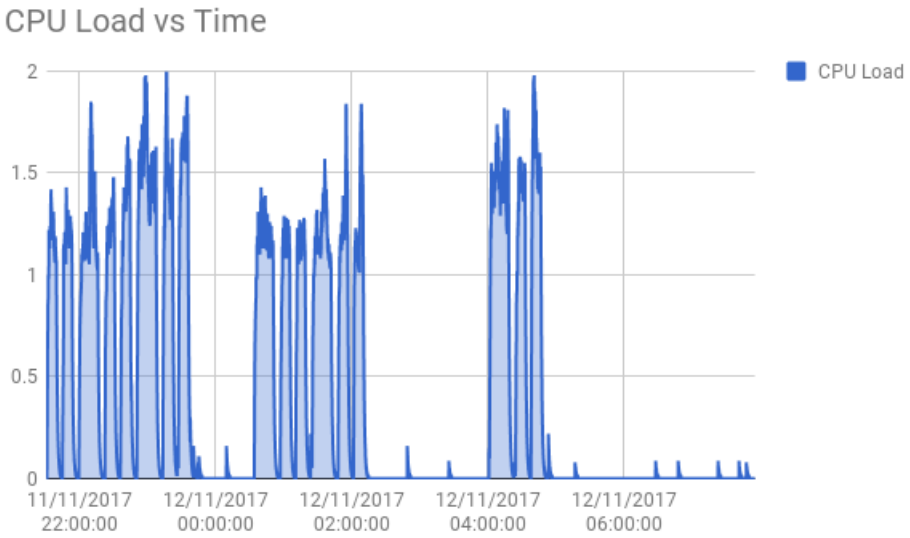
Charts

The four charts below show measurements over the same 11 hour period and plots each measurement (y-axis) against the time period (x-axis).

Again, the cpu load shows various periods of increased load and also troughs within high load periods (around 1.5 -2). Interesting, as we're now plotting cpu utilisation data alongside load data, we can relate this graph to the cpu utilisation graph and see that the cpu is fully utilised when the cpu is overloaded.

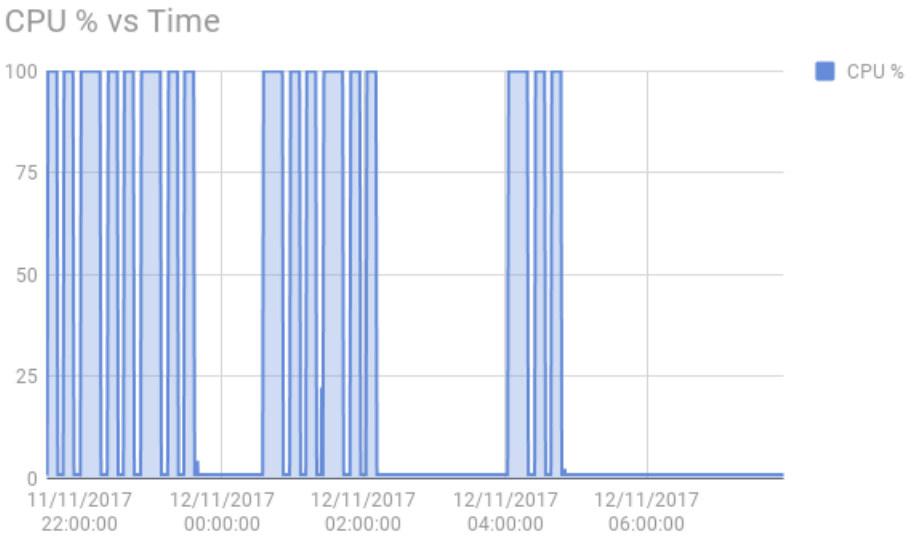
At around 01:25, the memory usage and network usage both spiked drastically at the same time. This could be a local program starting up and pinging and external service rapidly. The final chart displays the network and memory usage during this specific time period.

CPU Load vs Time



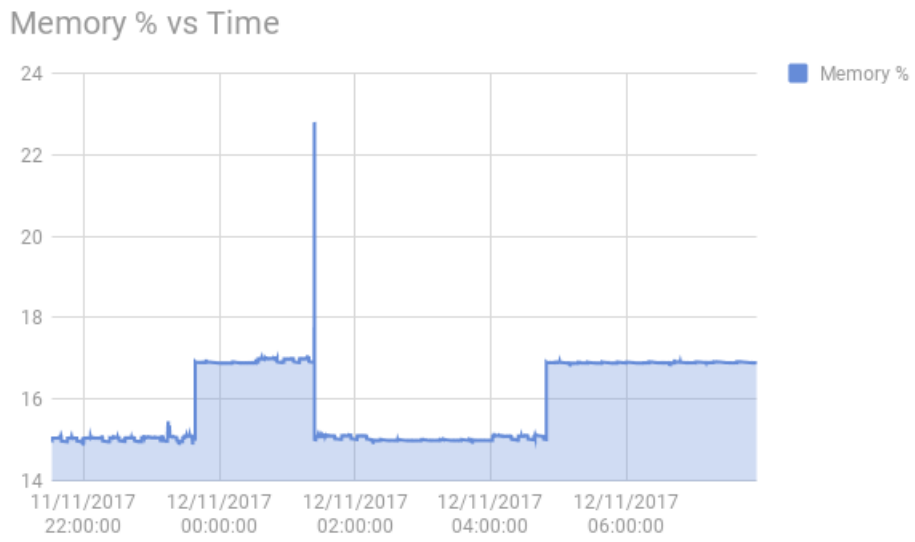

```
Measurements #: 7485
Mean: 0.41
Variance: 0.36
Standard deviation: 0.60
95% confidence interval: [0.40, 0.42]
99% confidence interval: [0.39, 0.42]
```

CPU % vs Time



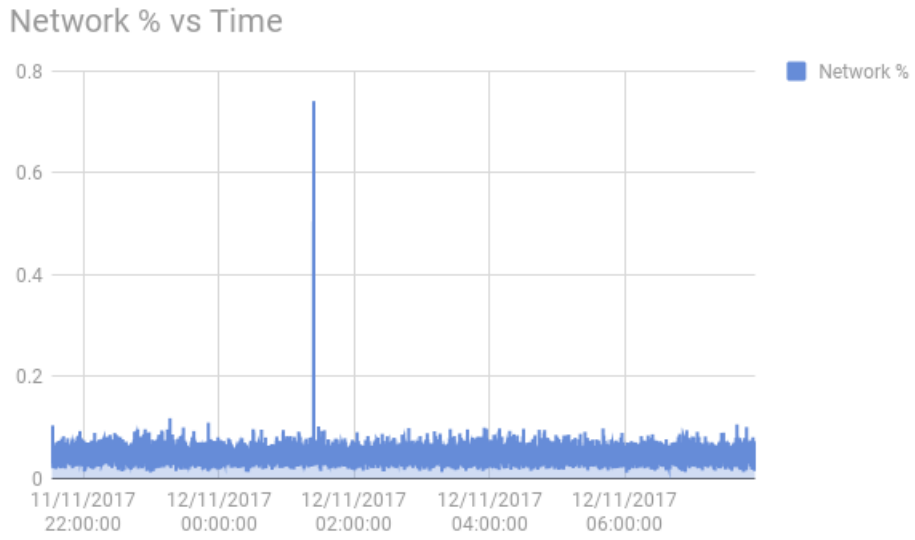

```
Measurements #: 7485
Mean: 30.38
Variance: 1973.74
Standard deviation: 44.43
95% confidence interval: [29.54, 31.23]
99% confidence interval: [29.38, 31.39]
```

Memory % vs Time




```
Measurements #: 7485
Mean: 15.91
Variance: 0.90
Standard deviation: 0.95
95% confidence interval: [15.89, 15.93]
99% confidence interval: [15.89, 15.93]
```

Network % vs Time




```
Measurements #: 7484
Mean: 0.05
Variance: 0.00
Standard deviation: 0.02
95% confidence interval: [0.05, 0.05]
```

```
99% confidence interval: [0.05, 0.05]
```

Unusual Memory / Network 01:25:00 Spike