Using sar2xls to create an xls formatted spreadsheet from a sar text file

The Perl script sar2xls can be used to create an xls formatted Excel/LibreOffice spreadsheet from a sysstat sar text output file (note: not the sar binary data file).

Installing

The script uses the Perl Spreadsheet::WriteExcel module to create the spreadsheet, and thus can be used from OS X, Windows or Linux as long as this and its required Perl modules are installed:

- OLE-Storage_Lite
- Parse-RecDescent
- Spreadsheet-WriteExcel

Installing sar2xls on Red Hat Enterprise Linux 6

The pre-requisite Perl modules are in the EPEL repository (http://fedoraproject.org/wiki/EPEL). Once the EPEL repository is added to your Red Hat Enterprise Linux 6 system, these Perl modules can be installed with the single command:

```
yum install perl-Spreadsheet-WriteExcel
```

Now download the sar2xls file from https://github.com/pemcg/sar2xls and copy it to a directory on your $PATH, or to a local working directory.

Installing sar2xls on Windows

Download and install the appropriate ActivePerl MSI installer bundle for the running version of Windows from http://www.activestate.com/activeperl/downloads

If required, set the `http_proxy` variable in accordance with http://docs.activestate.com/activeperl/5.10/faq/ActivePerl-faq2.html#ppm_and_proxies

Then from a command prompt:

```
ppm install Spreadsheet-WriteExcel
```

Now download the sar2xls file from https://github.com/pemcg/sar2xls and copy it to a directory on your $PATH, or to a local working directory.

Installing sar2xls on Mac OS X

The pre-requisite Perl modules are available from CPAN. Its recommended to set FTP into passive mode so as to avoid having any troubles due to CPAN not being able to download from repositories via ftp.

```
export FTP_PASSIVE=1
```

Then to run CPAN (as root):

```
sudo perl -MCPAN -e "shell"

cpan[1]> install Spreadsheet::WriteExcel
```

Now download the sar2xls file from https://github.com/pemcg/sar2xls and copy it to a directory on your $PATH, or to a local working directory.

Usage

The script is run as follows:

```
usage: sar2xls [-f sar_file | -a] [-t threshold_file] [-D directory] [-b begin_time -e
end_time] [-o output_file] [-m] [-V]

Where:
    -V prints the version of sar2xls
    -f Specifies the sar text file to process (excluding the directory path)
    -a Specifies all sar files in current or specified folder
    -t Optional threshold file for colour-coding cells
    -D Optional directory to use for input and output (default is the current directory)
    -b Begin Time
    -e End Time
    -m Indicates to break out per-CPU stats rather than the average for all CPUs
    -o Specifies an optional output file name (default is hostname-date)

     e.g.  sar2xls -f sar05 -D /path/to/sarfile -b 10:22:30 -e 10:45:00
           sar2xls -a -D /var/log/sa
```

Note that -f and -a are mutually exclusive: -a will recognise and process all sar files in the specified (or default) directory.
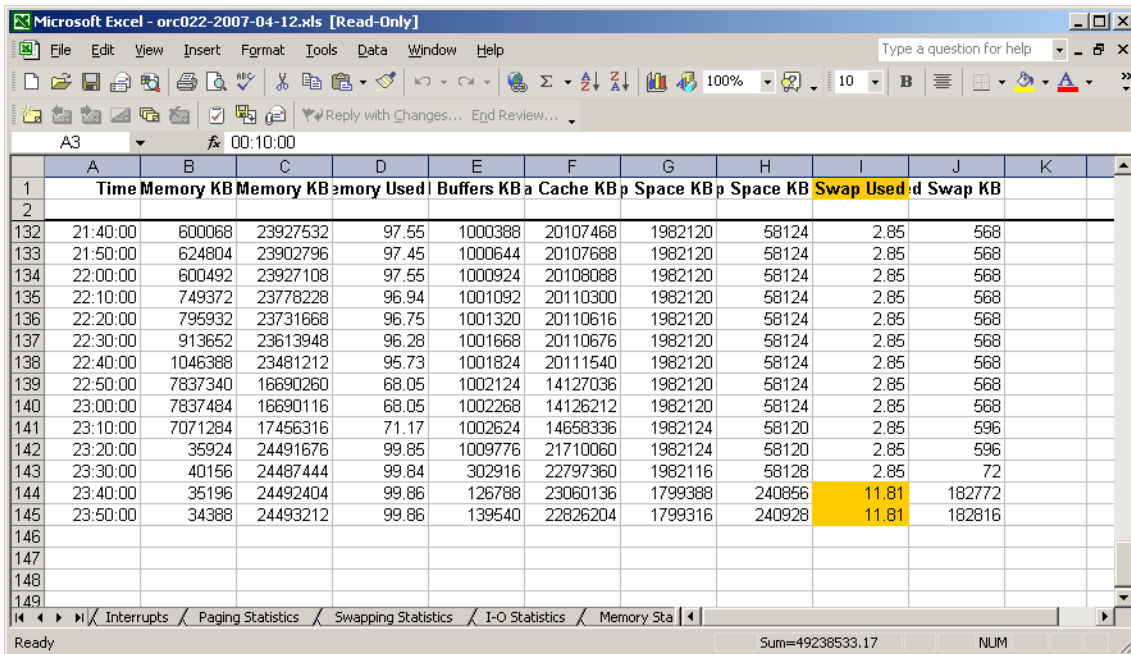
Time Slices

If we only wish to examine a particular time interval in the sar file, we can specify this using the begin and end (–b and –e) switches, i.e.

```
sar2xls -f sar_file -b 13:30:00 -e 14:00:00
```

Thresholds File

An optional threshold file can be specified on the command line, and is used to specify values to watch for, and colour-code a cell accordingly, i.e.

The file is formatted as follows:

```
[Amber]
parameter=value
parameter=value
[Red]
parameter=value
parameter=value

...

i.e.

[Amber]
%utilisation=50
rd_sec/s=61035
wr_sec/s=61035
%swpused=10
[Red]
%utilisation=70
rd_sec/s=122070
wr_sec/s=122070
%swpused=30
```

Each parameter should be specified exactly as it appears in the sar file, and the value can be positive or negative to specify a maximum or minimum value. Comment lines starting with a # or blank lines are ignored.

i.e.

```
%utilisation=50
```

would flag when CPU utilisation rose above 50%, and

```
kbmemfree=-10000
```

would flag when free memory sank below 10000KB

Device Mapping File

By default sar uses device major,minor IDs to identify devices in the device I/O section:

```
00:10:01     dev8-224       0.87       0.53       6.40
00:10:01     dev8-240       0.87       0.53       6.40
00:10:01      dev65-0      25.31     201.33       0.50
00:10:01       dev2-0       0.00       0.00       0.00
00:10:01       dev9-0       0.00       0.00       0.00
00:10:01     dev253-0       0.34       0.00       2.75
00:10:01     dev253-1      13.73       0.03     109.77
00:10:01     dev253-2       0.15       0.13       1.17
00:10:01     dev253-3       2.42       0.05      19.32
00:10:01     dev253-4       0.00       0.00       0.00
00:10:01     dev253-5       0.06       0.12       0.47
00:10:01       dev7-0       0.00       0.00       0.00
00:10:01       dev7-1       0.00       0.00       0.00
```

which can make it quite difficult to identify which actual device the statistics are referring to. sar2xls.pl will use an (optional) per-system device name mapping file if found in the current working directory, to map the major,minor device name to something more meaningful. The mapping file name must be `hostname -s`_dev_map, i.e. xxx020_dev_map, and the format of the mapping file is an follows:

```
[monitor2]
dev10-63 = mapper~control
dev253-0 = mapper~VolGroup00-LogVol02
dev253-1 = mapper~VolGroup00-LogVol01
dev253-2 = mapper~VolGroup00-LogVol00
dev8-0 = dev~sda
dev8-1 = dev~sda1
dev8-16 = dev~sdb
dev8-17 = dev~sdb1
dev8-2 = dev~sda2
dev9-0 = dev~md0
…
```

If the mapping file is found, sar2xls will translate device names according to the entry in the mapping file, i.e.

Unfortunately Excel does not permit '/' characters in the names of worksheets, so "/dev/sda" would not be allowed.

A mapping file can be created using the *mk_dev_map.pl* script