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.

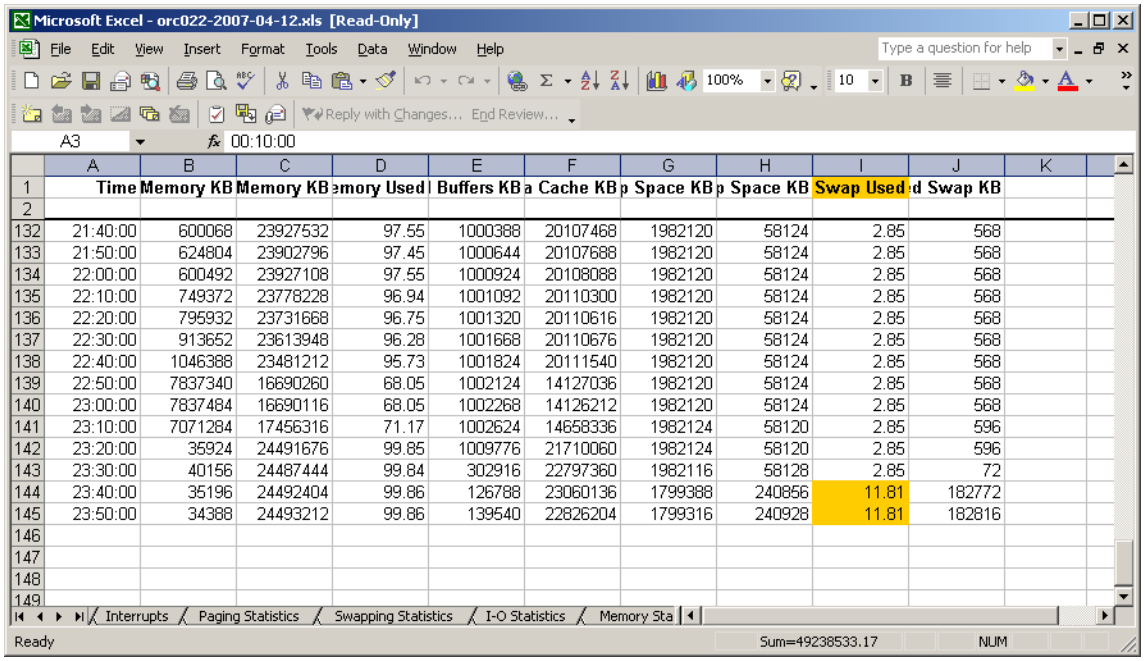
Time Slices

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.



	A	B	C	D	E	F	G	H	I	J	K
1	Time	Memory KB	Memory KB	emory Used	Buffers KB	a Cache KB	p Space KB	p Space KB	Swap Used	d Swap KB	
2											
132	21:40:00	600068	23927532	97.55	1000388	20107468	1982120	58124	2.85	568	
133	21:50:00	624804	23902796	97.45	1000644	20107688	1982120	58124	2.85	568	
134	22:00:00	600492	23927108	97.55	1000924	20108088	1982120	58124	2.85	568	
135	22:10:00	749372	23778228	96.94	1001092	20110300	1982120	58124	2.85	568	
136	22:20:00	795932	23731668	96.75	1001320	20110616	1982120	58124	2.85	568	
137	22:30:00	913652	23613948	96.28	1001668	20110676	1982120	58124	2.85	568	
138	22:40:00	1046388	23481212	95.73	1001824	20111540	1982120	58124	2.85	568	
139	22:50:00	7837340	16690260	68.05	1002124	14127036	1982120	58124	2.85	568	
140	23:00:00	7837484	16690116	68.05	1002268	14126212	1982120	58124	2.85	568	
141	23:10:00	7071284	17456316	71.17	1002624	14658336	1982124	58120	2.85	596	
142	23:20:00	35924	24491676	99.85	1009776	21710060	1982124	58120	2.85	596	
143	23:30:00	40156	24487444	99.84	302916	22797360	1982116	58128	2.85	72	
144	23:40:00	35196	24492404	99.86	126788	23060136	1799388	240856	11.81	182772	
145	23:50:00	34388	24493212	99.86	139540	22826204	1799316	240928	11.81	182816	
146											
147											
148											
149											

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

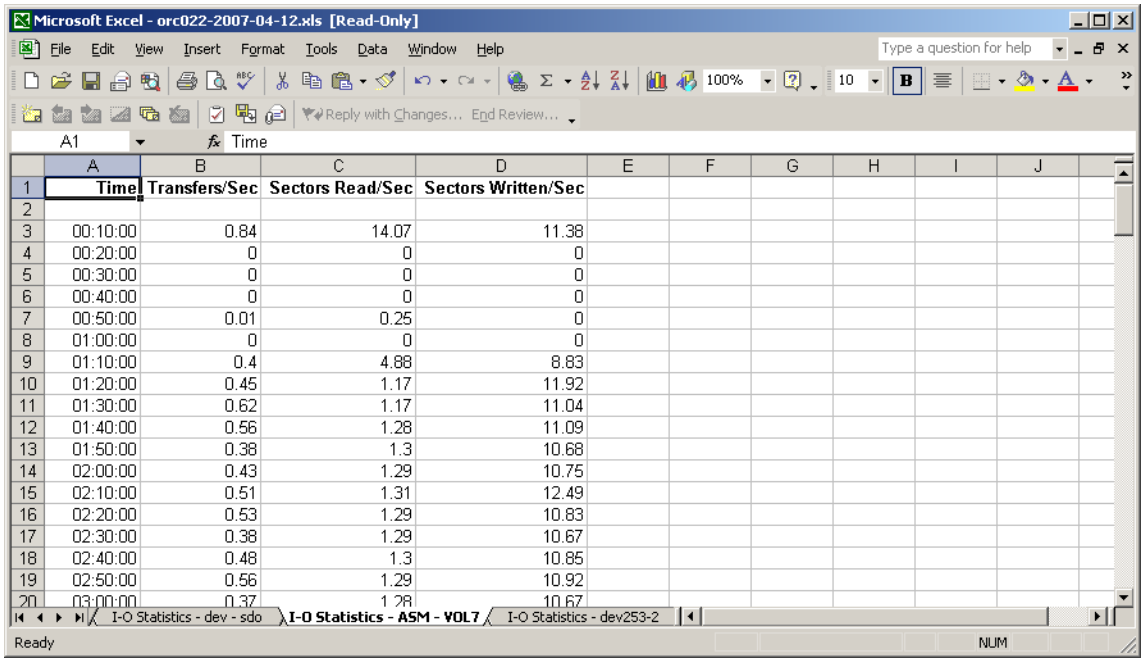
By default sar uses device major,minor IDs to identify devices in the device I/O section:

```
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:

```
[hostname]
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.



	A	B	C	D
1	Time	Transfers/Sec	Sectors Read/Sec	Sectors Written/Sec
2				
3	00:10:00	0.84	14.07	11.38
4	00:20:00	0	0	0
5	00:30:00	0	0	0
6	00:40:00	0	0	0
7	00:50:00	0.01	0.25	0
8	01:00:00	0	0	0
9	01:10:00	0.4	4.88	8.83
10	01:20:00	0.45	1.17	11.92
11	01:30:00	0.62	1.17	11.04
12	01:40:00	0.56	1.28	11.09
13	01:50:00	0.38	1.3	10.68
14	02:00:00	0.43	1.29	10.75
15	02:10:00	0.51	1.31	12.49
16	02:20:00	0.53	1.29	10.83
17	02:30:00	0.38	1.29	10.67
18	02:40:00	0.48	1.3	10.85
19	02:50:00	0.56	1.29	10.92
20	03:00:00	0.37	1.28	10.67

Unfortunately Excel does not permit '/' characters in the names of worksheets, so "/dev/sda" would not be allowed. Use something meaningful like dev~sda instead.

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