

Solaris Configuration and Performance Survey Tools

Version 13
June 29th, 2021

Table of Contents

1 Sysinfo – Collect Solaris/SPARC Configuration Information.....	2
1.1 Description.....	2
1.2 Compatibility.....	2
1.3 Options.....	2
1.4 Dry-run Mode.....	2
1.5 Removing Sensitive Information.....	3
1.6 Basic Performance Data.....	3
1.8 Running the Script.....	4
1.9 Output.....	4
2 Performance Data Collection Script.....	6
2.1 Compatibility.....	6
2.2 Run in Global Zone.....	6
2.3 Avoid Configuration Changes During Collection.....	6
2.4 Prerequisite: sar.....	6
2.5 Description.....	7
2.7 Options.....	8
2.8 When and How Long to Run the Script.....	8
2.9 Filesystem Free Space Check.....	8
2.10 Make the Script File Executable.....	9
2.11 Running the Script Interactively.....	9
2.12 Running the Script Under nohup Control.....	9
2.13 Output.....	10
2.14 Analysis.....	10

1 Sysinfo – Collect Solaris/SPARC Configuration Information

1.1 Description

`sysinfo` (current version: `sysinfo-v112.sh`) is a relatively short (currently ~700 line) Bourne Shell script designed to collect configuration information from a SPARC system running Solaris or SunOS. It runs a series of commands that display configuration information and redirects the output of the commands to an output file which you then send to Stromasys.

Please note that the version number, currently “1.12” can change. Below the script is simply referred to as `sysinfo.sh`.

This script *must* run with super-user (“root”) privileges.

1.2 Compatibility

`sysinfo.sh` can be run on SunOS 4.1.x (Solaris 1.x) and all SunOS 5.x (Solaris 2.x, 7, 8, 9, 10 and 11.x) versions.

1.3 Options

The script has several options to control its behavior:

<code>-n</code>	Directs the script to perform a “dry-run”, meaning the commands are not actually executed
<code>-p</code>	Disables the collection of some basic performance statistics. On Solaris, <code>vmstat</code> and <code>sar</code> On SunOS, <code>vmstat</code> , <code>iostat</code> and <code>mpstat</code>
<code>-m <minutes></code>	Specifies how many minutes to collect the performance data. Example: <code>-m 5</code> (not relevant if <code>-p</code> is used)
<code>-s <unique-name></code>	Enables “sanitization” and provides an anonymous, but unique hostname. Example: <code>-s host1</code>
<code>-c</code>	Disables compression of the output file (<code>gzip</code> if present, otherwise <code>compress</code>)
<code>-h</code>	Print a short summary of the options

1.4 Dry-run Mode

The “dry-run” mode (`-n`) will produce an output file with only the commands that would have been executed, but without actually executing them. Dry-run mode must execute two commands in order to be able to proceed. These two commands are:

- `uname -r` (to determine if the OS is Solaris or SunOS)
- `echo | format` (to print a list of disks attached to the system)

1.5 Removing Sensitive Information

If you are concerned about divulging sensitive information about the system being surveyed, you can go through the output file, which is simply text and possibly redact it according to your needs. In particular, the `ifconfig` command is used to collect configuration details for the network interfaces, including IP-addresses.

Alternatively, you can use the “sanitize” option (`-s <unique-name>`), which automatically performs the following replacements in the output file:

Value	Replaced With
hostname	<hostname>
host-ID	<hostid>
IP-address	XXX.XXX.XXX.XXX
MAC-address	XX:XX:XX:XX:XX:XX
hostnames of remote systems	<remote>

The localhost address, `127.0.0.1`, is not replaced.

The hostnames in the output of the `mount`, `df`, and `cat /etc/vfstab` commands are replaced with the string “<remote>”.

The option to enable sanitization is `-s <name>`, where `<name>` is some unique name to associated with the system being surveyed. We need this to be able to discuss a particular system later. You should keep some internal table with a mapping of unique-name to real hostname.

When the script is finished, it produces two files: one with sanitized output, which you should send to Stromasys, and one with unsanitized data, which you should keep in case we have a question about something that has been sanitized. The script prints the full paths to both the files at the end of the run.

We recommend performing sanitization by hand if possible, because the automatic feature can remove useful diagnostic information.

1.6 Basic Performance Data

The `sysinfo` script by default collects some very basic performance data, unless you specify the `-p` option to disable this. Also by default, it collects the performance data for 1 minute. This can be changed by supplying the `-m <minutes>` option. If possible, please run the script at a time of peak workload, and consider collecting several (~5) minutes of performance-data.

For comprehensive collection of performance data, please use the `collect-perfdata.ksh` script we provide for that purpose (below).

1.7

1.8 Running the Script

Please run `sysinfo.sh` at a time when the system is under load if you intend to allow it to collect performance statistics. To run, as the “root” user, change to the directory where `sysinfo.sh` is located and type:

```
# ./sysinfo.sh
```

possibly with the `-n`, `-p`, `-m <minutes>`, `-s <name>` or `-c` flags, according to your needs.

Example output

```
## running at Monday  2 March 2020 10:29:41 GMT ##
## sysinfo version: 1.3.12 ##
##-----
# CMD: hostname
##-----
myhostname

##-----
# CMD: uname -a
##-----
SunOS myhostname 5.10 Generic_139555-08 sun4v sparc SUNW,SPARC-Enterprise-T5220

##-----
# CMD: cat /etc/release
##-----
                        Solaris 10 10/08 s10s_u6wos_07b SPARC
                Copyright 2008 Sun Microsystems, Inc.  All Rights Reserved.
                  Use is subject to license terms.
                  Assembled 27 October 2008

...and so on...
```

Each command executed is shown in a header and the output of that command follows the header.

1.9 Output

The script produces a file with the name:

```
/tmp/sysinfo-<hostname>-<time-stamp>.out.gz
```

where `<hostname>` is the non-fully-qualified-domain-name of the system on which the script is run, and `<time-stamp>` has the format `YYYY-MM-DD-HHSS`. If `gzip` is not available on the system, the file will be compressed with `compress` and will have the extension “`.Z`” instead of “`.gz`”. If the `-c` flag is used, the output file will not be compressed.

If sanitization was enabled, the sanitized output will be in the file:

```
/tmp/sysinfo-<unique-name>-<time-stamp>.out.gz
```

and the unsanitized output will be in the file:

```
/tmp/sysinfo-<unique-name>-<time-stamp>-unsanitized.out.gz
```

Keep in mind that files in the `/tmp/` directory may be deleted automatically after a time and almost certainly if the system is restarted, so the sysinfo result files should be moved somewhere safe.

2 Performance Data Collection Script

`collect-perfdata-v16.ksh` is a KornShell script designed to collect performance statistics over a certain period of time (collection-period).

The script must be started only **once** at the beginning of the collection-period and run until the end of the collection-period. The `-r <run-time>` option must be used to specify the length of the collection-period. The script will refuse to run if it detects a script instance already running.

The version number (currently 16) of the script can change. Henceforth the script will be referred to as `collect-perfdata.ksh`.

This script *must* run with super-user (“root”) privileges.

2.1 Compatibility

`collect-perfdata.ksh` can be run on SunOS 5.8 and newer (Solaris 8, 9, 10, and 11.x) versions.

2.2 Run in Global Zone

You must run `collect-perfdata.ksh` in the global zone on a system with multiple zones. The script will refuse to run in a non-global zone.

`collect-perfdata.ksh` collects zone CPU utilization data for all zones when running in the global zone. On Solaris 11 zone-specific network utilization data is also available.

On systems with zones, run `collect-perfdata.ksh` in the global zone only.

2.3 Avoid Configuration Changes During Collection

`collect-perfdata.ksh` takes a snapshot of certain configured subsystems (network, zone) when it starts. If NICs or zones disappear during performance-data collection, it causes errors when we attempt to analyze the data.

Please do not stop any zones or change the network configuration while `collect-perfdata.ksh` is running.

2.4 Prerequisite: sar

The “sar” (System Activity Reporter) utility must be installed on the system from which data will be collected. sar is part of the base-OS distribution. The package name is “SUNWaccu”. On Solaris 8 it can be found on the 2nd companion CD. On Solaris 10 on the installation DVD. To install the sar utility, cd to the directory which contains the `SUNWaccu` subdirectory, and type (as “root”):

```
# pkgadd -d . SUNWaccu
```

and then confirm the installation.

The script checks if the sar utility is available. If not, it prints an error and exits.

2.5 Description

The script runs the `sar` command continuously and runs the `uptime`, `ps`, `netstat`, `iostat` and `mpstat` commands periodically over the entire time it is running. Depending on the Solaris version, it may also run the `dladm` or `kstat -c net -m <mod>` commands. On Solaris 11 it also runs the `zonestat` command.

By default the script raises its priority to the maximum possible (-20) in order to increase the quality of the data gathered. This can be disabled using the **-p** switch.

2.6

2.7 Options

The script takes the following options:

Option	Description
-t <temporary-directory>	Where <temporary-directory> is the path to a directory, such as /var/tmp
-r <run-time>	Specify run-time using “w”, “d”, “h”, “m”, and “s”. Example: “3d6h20m” (3 days, 6 hours and 20 minutes) Maximum allowed value: 2 weeks
-i <collection-interval>	Sets the collection-interval. The default is 5 seconds (strongly recommended) (maximum allowed value: 30 seconds)
-p	Disable raising priority of process
-s	Do not check if available disk space is sufficient
-v	Increase verbosity
-V	Prints the version of the script
-h	Prints a short summary of the options
-n	Do not combine result files in a tar archive

2.8 When and How Long to Run the Script

The script should be run over a period of many hours such that at the beginning the workload on the system is lowest, continue to run during the time when the system reaches maximum workload, and run until the system workload has decreased significantly, if possible until the workload is lowest again. For example, if a system has highest workload at 2 PM, the script could be started at 8 AM, and run until 8 or 9 PM, or later, depending on how the workload decreases.

Your period of peak workload may be every day, once a week, or at the end of every month or quarter, for example at the end of some billing period. Only you can know when your peak workload occurs. This means you must wait until the peak workload to collect performance data.

If you do not collect data during peak workload, our analysis will not be correct.

As the script can generate a fair amount of data – possibly several gigabytes. By default, it writes the output files to /var/tmp/. You can specify an alternate destination directory using
-t /path/to/directory.

Please do not change the collection interval (default is 5 seconds) unless you have a very good reason and have spoken to us about it.

2.9 Filesystem Free Space Check

The script collects in a test mode for 10 seconds during initialization in order to determine how much space will be occupied by the data-files at the end of collection. It checks the free space of the target filesystem and if the estimated final size of the data is more than 90% of the available free space of the filesystem, it will abort execution. The -s flag will disable this check.

2.10 Make the Script File Executable

Run the following command to make the script file executable:

```
# chmod 755 collect-perfdata.ksh
```

2.11 Running the Script Interactively

To run the script, you must use the `-r <run-time>` option to tell the script how long to run. At the end of the time specified, the script will automatically terminate itself and generate a tar-archive containing the individual result files. The `<run-time>` is limited to a maximum of 2 weeks.

The format of `<run-time>` is a series of `<number><unit>` pairs, where `<number>` is one or more digits, and `<unit>` is one of “w”, “d”, “h”, “m”, or “s”, which mean weeks, days, hours, minutes, and seconds. There must be no white-space in the `<run-time>` expression.

Examples:

```
-r 1w1d    (one week and 1 day)
-r 8d      (eight days, same as previous)
-r 2d6h    (2 days and 6 hours)
-r 54h     (54 hours, same as previous).
```

This method requires that you establish a terminal session to the system on which the script will be run. This session can be remote, via telnet or ssh, or on the console directly. The session must exist for the time the script will be run – probably many hours. If there is a risk that the session might be disconnected while the script is running, please refer to the method below (“Running the Script Under nohup Control”).

As “root”, please change directory to where the script is located and type:

```
# ./collect-perfdata.ksh -r <run-time> [-t /target/directory/]
```

possibly including the `-t` and or `-i` flags.

The script will automatically stop at the end of the collection-period. You can type **Ctrl-C** (hold the “CONTROL” key and press the “C” key once and release immediately) to terminate the script before the end of the collection-period.

WARNING: Do NOT use `kill -9 <pid>` to stop the script. If you do this, subprocesses such as “mpstat” and “iostat” may continue to run indefinitely, and you will prevent the script from exiting cleanly.

2.12 Running the Script Under nohup Control

In order to mitigate the risk of having the script terminated prematurely should the interactive session be lost, it is possible to run it under the control of `nohup`, as follows:

```
# nohup ./collect-perfdata.ksh -r <run-time> [-t /target/directory] &
```

This will run the script in the background, and with the help of `nohup` it should survive the loss of the interactive session.

In this case, the informational output from the script that would normally be printed to standard-output is directed instead to the file `nohup.out`.

The script will write its Process-ID to the file `/tmp/collect-perfdata-PID.txt`. To terminate it before the end of the collection-period, use the following command:

```
# kill -TERM `cat /tmp/collect-perfdata-PID.txt`
```

Please note that the characters surrounding the text “`cat /tmp/collect-perfdata-PID.txt`” are *back-ticks* (```) and *not* apostrophes (`'`).

Alternatively, you can simply run `cat /tmp/collect-perfdata-PID.txt`, and then `kill -TERM <PID>`, where `<PID>` is the result of the `cat` command.

Be sure to wait until all `collect-perfdata.ksh` processes have exited. This can be checked by typing:

```
# ps -ef | grep collect-perfdata | grep -v grep
```

2.13 Output

The script will produce three files in `/var/tmp`, or the directory specified when the script was run. The files produced are:

- `perfdata-<hostname>-misc-<time-stamp>.txt`
- `perfdata-<hostname>-sar-<time-stamp>.txt`
- `perfdata-<hostname>-mpstat-<time-stamp>.txt`
- `perfdata-<hostname>-zonestat-<time-stamp>.txt`
- `perfdata-<hostname>-timestamp-<time-stamp>.txt`

`<time-stamp>` has the format `YYYY-MM-DD-HHmm`. The `zonestat` file will only be produced on Solaris-11 systems when running the script in the global zone. These files are then added to a tar archive which is finally compressed. The name of the compressed tar archive is:

```
perfdata-<hostname>-<time-stamp>.tar.gz
```

Starting with version 14, `collect-perfdata.ksh` uses a log-file to keep diagnostic information about its execution. The log-file name is `perfdata-<hostname>-<time-stamp>.log`.

Please send both the tar archive and the log-file to us.

2.14 Analysis

The script allows Stomasys to plot performance data graphically. Here is an example:

