Stress Test CPU and Memory (VM) On a Linux / Unix With Stress-ng

last updated July 18, 2018 **in FreeBSD, Hardware, Linux, OpenBSD, UNIX**

I want test my Linux, OpenBSD, FreeBSD and Unix-like server entirely for high load and monitoring the health under stress. How can I stress out my CPU, memory, I/O, and disk stress and more with stress test tool on a Linux or Unix-like systems?

UNIX

A sysadmin can try out any one of the following tool to put given subsytems under a specified load.[donotprint][/donotprint]One can stress test CPU on Linux other subsystem. Instances in which this is useful include those in which a system administrator wishes to perform tuning activities, a kernel or libc programmer wishes to evaluate denial of service possibilities, test your systems entirely on high load and monitoring the health and more. This is also useful for sysadmin, system builders, and overclockers who want to test their hardware under high load and monitor stability and thermal environment.

Tools To Stress Test CPU and Memory (VM) On a Linux / Unix

1. **`stress`** : It is a simple workload generator for POSIX systems. It imposes a configurable amount of CPU, memory, I/O, and disk stress on the system. It is written in C, and is free software licensed under the GPLv2. It is not a benchmark, but is rather a tool designed
2. **`stress-ng`** : It is an updated version of stress tool and it will stress test a server for the following features:
    1. CPU compute
    2. Cache thrashing
    3. Drive stress
    4. I/O syncs
    5. VM stress
    6. Socket stressing
    7. Context switching
    8. Process creation and termination
    9. It includes over 60 different stress tests, over 50 CPU specific stress tests that exercise floating point, integer, bit manipulation and control flow, over 20 virtual memory stress tests.

Warning: Running the following tools with root privileges is recommended to avoid out of memory and other errors. Also, note that tools will stress out your server resources quickly so use the following command judiciously.

Getting started with stress tool on Linux

This program is supposed to be easy to use and recommended for new sysadmins. The tool is known to work on x86 Linux and FreeBSD/OpenBSD, powerpc AIX and Linux, SPARC Solaris, Compaq Alpha Tru64 UNIX, and many others.

Install stress package on a Linux

You can install stress as part of the Linux or Unix distribution.

INSTALL STRESS ON A CENTOS, RHEL, AND FEDORA LINUX

First, turn on EPEL repo and then type the following yum command to install the same:

```
sudo yum install stress
```

Sample outputs:

Fig.01: Installing stress on a RHEL/CentOS/Fedora Linux

INSTALL STRESS ON A DEBIAN AND UBUNTU LINUX

Type the following apt-get command to install the same:

```
apt-get install stress
```

Sample outputs:

Fig.02: Install stress tool on a Debian/Ubuntu Linux

INSTALL STRESS ON A FREEBSD UNIX SYSTEM

Type the following pkg command to install the stress tool using binary method:

```
pkg install stress
## OR ##
pkg install sysutils/stress
```

Sample outputs:

Fig.03: FreeBSD installing stress tool

INSTALL STRESS ON A OPENBSD UNIX SYSTEM

Type the following pkg_add command to install the stress tool using binary method:

```
## if installpath not set in  /etc/pkg.conf as follow ##
## installpath =   http://mirror.esc7.net/pub/OpenBSD/%c/packages/%
a/ use ##
## PKG_PATH  ##
export PKG_PATH=http://ftp.usa.openbsd.org/pub/OpenBSD/`uname -r`/p
ackages/`arch -s`
pkg_add stress
```

How do I use stress tool on Linux?

First, note down the current system load averages by typing the following command:

```
# uptime
```

Next, run any one of the following command to see load on screen:

```
# watch uptime
```

OR use tload command:

```
# tload
```

The syntax is as follows:

```
stress [OPTION]
## Stress using CPU-bound task
stress -c 4
## Stress using IO-bound task
stress -i 2
```

For example, a load average of four is imposed on the system by specifying two CPU-bound processes, one I/O-bound process, and one memory allocator process as follows:

```
# uptime

# stress -c 2 -i 1 -m 1 --vm-bytes 128M -t 10s

# uptime
```

Sample outputs:

Fig.04: A load average of four is imposed on the system

Where,

- `-c 2` : Spawn two workers spinning on sqrt()
- `-i 1` : Spawn one worker spinning on sync()
- `-m 1` : Spawn one worker spinning on malloc()/free()
- `--vm-bytes 128M` : Malloc 128MB per vm worker (default is 256MB)
- `-t 10s` : Timeout after ten seconds
- `-v` : Be verbose

Getting started with stress-ng on Linux and Unix

The `stress-ng` tool will stress test a Linux computer system in various selectable ways.

Install stress-ng on a Linux or Unix-like systems

Type the following command to download stress-ng tarball using the wget command:

```
$ cd /tmp
$ wget http://kernel.ubuntu.com/~cking/tarballs/stress-ng/stress-ng-
0.09.34.tar.xz
```

Untar tar ball, enter:

```
$ tar xvf stress-ng-0.09.34.tar.xz
```

Compile stress-ng, run:

```
$ cd stress-ng-0.09.34
$ make
```

Sample outputs:

```
make -f Makefile.config
make[1]: Entering directory '/tmp/stress-ng-0.09.34'
autoconfig: using -lcrypt
autoconfig: using pthread spinlock
autoconfig: using -lrt
autoconfig: using -lz
autoconfig: using -ldl
autoconfig: using -lpthread
autoconfig: using wchar.h
autoconfig: using grp.h
autoconfig: using sys/xattr.h
autoconfig: using sys/syscall.h
.....
...
.....
CC mwc.c
CC parse-opts.c
CC out-of-memory.c
CC net.c
CC sched.c
CC setting.c
CC thermal-zone.c
CC shim.c
CC perf.c
CC thrash.c
CC time.c
CC stress-ng.c
CC stress-personality.c
LD stress-ng
make[1]: Leaving directory '/tmp/stress-ng-0.09.34'
```

You can also install it using snap command or apt-get command:

```
$ sudo apt install stress-ng
```

How do I use the stress-ng tool on Linux?

The syntax is:

```
stress-ng [options]
stress-ng -c 2
stress-ng -c 4 -t 10 -v
stress-ng -c 4 --metrics-brief
```

Examples

Always note down the output of uptime command before starting it:

```
uptime
```

Let us see some examples of stress-ng.

UNIX / LINUX CPU STRESS TEST

Let us start N workers exercising the CPU by sequentially working through all the different CPU stress methods:

```
uptime
stress-ng --cpu 4 --timeout 60s --metrics-brief
uptime
```

UNIX / LINUX CPU STRESS TEST

For disk start N workers continually writing, reading and removing temporary files:

```
stress-ng --disk 2 --timeout 60s --metrics-brief
```

One can pass the `--io N` option to the stress-ng command to commit buffer cache to disk:

```
stress-ng --disk 2 --io 2 --timeout 60s --metrics-brief
```

UNIX / LINUX MEMORY STRESS TEST

Let us populate memory. Use mmap N bytes per vm worker, the default is 256MB.

One can specify the size as % of total available memory or in units of Bytes, KBytes, MBytes and GBytes using the suffix b, k, m or g:

```
stress-ng --vm 2 --vm-bytes 1G --timeout 60s
```

The `--vm 2` will start N workers (2 workers) continuously calling mmap/munmap and writing to the allocated memory. Note that this can cause systems to trip the kernel OOM killer on Linux systems if not enough physical memory and swap is not available.

PUTTING IT ALL TOGETHER

To run for 60 seconds with 4 cpu stressors, 2 io stressors and 1 vm stressor using 1GB of virtual memory, enter:

```
stress-ng --cpu 4 --io 2 --vm 1 --vm-bytes 1G --timeout 60s --metrics-brief
```

Sample outputs:

Fig.05: stress-ng in action showing Stress Test CPU and Memory (VM) On a Linux

In this example, run 16 cpu stressors and stops after 900000 bogo operations:

```
stress-ng --cpu 16 --cpu-ops 900000
stress-ng --cpu 16 --cpu-ops 900000 --timeout 16
```

Sample outputs:

```
stress-ng: info:  [30367] dispatching hogs: 16 cpu
stress-ng: info:  [30367] successful run completed in 60.17s (1 min
, 0.17 secs)
```

To run 4 simultaneous instances of all the stressors sequentially one by one, each for 6 minutes and summaries with performance metrics at the end:

```
stress-ng --sequential 4 --timeout 6m --metrics
```

To run 2 FFT cpu stressors, stop after 5000 bogo operations and produce a summary just for the FFT results:

```
stress-ng --cpu 2 --cpu-method fft --cpu-ops 5000 --metrics-brief
```

To run cpu stressors on all online CPUs working through all the available CPU stressors for 2 hour:

```
stress-ng --cpu 0 --cpu-method all -t 2h
```

To run 2 instances of all the stressors for 10 minutes:

```
stress-ng --all 2 --timeout 10m
```

To run 128 stressors that are randomly chosen from all the available stressors:

```
stress-ng --random 128
```

To run 64 instances of all the different cpu stressors and verify that the computations are correct for 5 minutes with a bogo operations summary at the end:

```
stress-ng --cpu 64 --cpu-method all --verify -t 5m --metrics-brief
```

To run all the stressors one by one for 5 minutes, with the number of instances of each stressor matching the number of online CPUs:

```
stress-ng --sequential 0 -t 5m
```

To run all the stressors in the io class one by one for 1 minutes each, with 8 instances of each stressor running concurrently and show overall time utilisation statistics at the end of the run:

```
stress-ng --sequential 8 --class io -t 1m --times
```

SHOULD I RUN STESS-NG WITH ROOT ACCESS?

From the man page:

> Running stress-ng with root privileges will adjust out of memory settings on Linux systems to make the stressors unkillable in low memory situations, so use this judiciously. With the appropriate privilege, stress-ng can allow the ionice class and ionice levels to be adjusted, again, this should be used with care. However, some options do requires root privilege to alter various /sys interface controls. See stess-ng command man page for more info.

Conclusion

You just learned how to use the stess and stress-ng command to impose high CPU load on Linux and Unix-like system.

References

- Linux and Unix Test Disk I/O Performance With dd Command
- ~~stress home page~~ – Download source code and docs.
- stress-ng home page– Download source code and docs.

SHARE ON Facebook Twitter

The best massage gun
(according to Men's Health)
Looking to speed up growth and recovery time? This is the hottest trend among therapists and athletes.
Buy Now
KRAFTGUN

Posted by: Vivek Gite

The author is the creator of nixCraft and a seasoned sysadmin, DevOps engineer, and a trainer for the Linux operating system/Unix shell scripting. Get the **latest tutorials on SysAdmin, Linux/Unix and open source topics via** **RSS/XML feed** or weekly email newsletter.

Historical Comment Archive

16 comment

Jari Luukkonen January 29, 2015 at 10:05 am

you got typo

Install stress on a CentOS, RHEL, and Fedora Linux
First, turn on EPEL repo and then type the following yum command to install the same:

sudo yum install stree

> **nixCraft** January 29, 2015 at 11:10 am
>
> Thanks for the heads up!
>
> > **cvillepete** January 30, 2015 at 1:27 pm
> >
> > There are a few typos. Sign me up to review your articles if you need help. ;)

Colin King January 29, 2015 at 10:58 am

Thanks for the write up on stress-ng. I am still adding more stress features to stress-ng, but it is now almost “feature complete”. If one has ideas for new stressors please let me know!

> **IP3G** January 30, 2015 at 6:24 am
>
> Are you the author?
>
> > **Colin King** January 30, 2015 at 9:10 am

> Indeed I am.

Starlight86 February 3, 2015 at 2:26 pm

Thumb up for the work. @Colin

gae December 21, 2015 at 11:54 pm

This script generates a controlled CPU load on the selected core. Pretty simple and scientific solution. https://github.com/GaetanoCarlucci/CPULoadGenerator

Jamal Schmidtman January 14, 2016 at 4:23 pm

Nice article. Used this to get stress installed on my Centos system.Thank you!

Marcos February 11, 2016 at 4:45 pm

Hi, which software did you use to make the "Fig.05: stress-ng in action"?

I want to know hot to put arrows and comment in a image and all I know is using Paint-like softwares.

Nice tutorial.

pratbid March 7, 2016 at 11:29 am

hey
i need tools that could induce load on system CPU ,i/o etc in any platform. if you know please suggest me

Anthony March 15, 2016 at 5:11 am

I'm not seeing the package in the EPEL el7 repo.

> **Aaron** July 9, 2016 at 11:56 am
>
> Its in RepoForge in EL7
> http://repoforge.org/use/
>
> yum search stress
> — Output omitted
> stress.x86_64 : Tool to impose stress on a POSIX-compliant operating system
> fio.x86_64 : I/O benchmark and stress/hardware verification tool
> httpress.x86_64 : HTTP stress & benchmark utility

Ian June 9, 2016 at 8:21 am

To put load on the cpu without any third party tools:
yes>/dev/null

Christian Matsoukis November 10, 2016 at 2:49 pm

For new users to Linux, when you run the "make" command run "make install" after it if you are having issues getting "stress-ng" to work.

JoeG December 15, 2016 at 4:59 pm

You should make a small container and a web interface that can detect the CPU/RAM of the VM as well as the disk. TrueNAS/FreeNAS plugin as a jail/container would be great as they really need a default testing tool that all users can quickly install/run and test as well as share the results.

Still, have a question? Get help on our forum!

Tagged as: CentOS, fedora, Linux, linux operating systems, red hat enterprise, UNIX, Advanced