Linux Load Averages: Solving the Mystery

Load Average in LINUX

Autour: M.M.Mirzaei

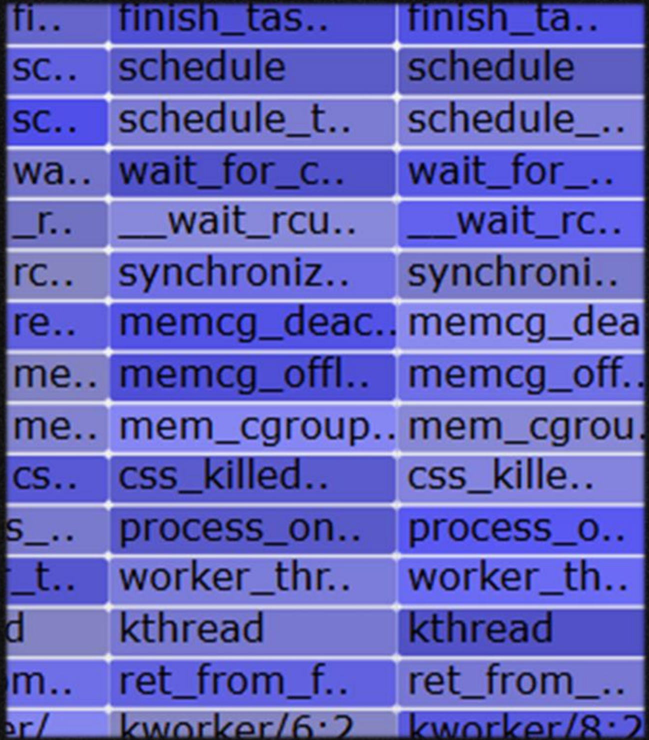
Table of contents

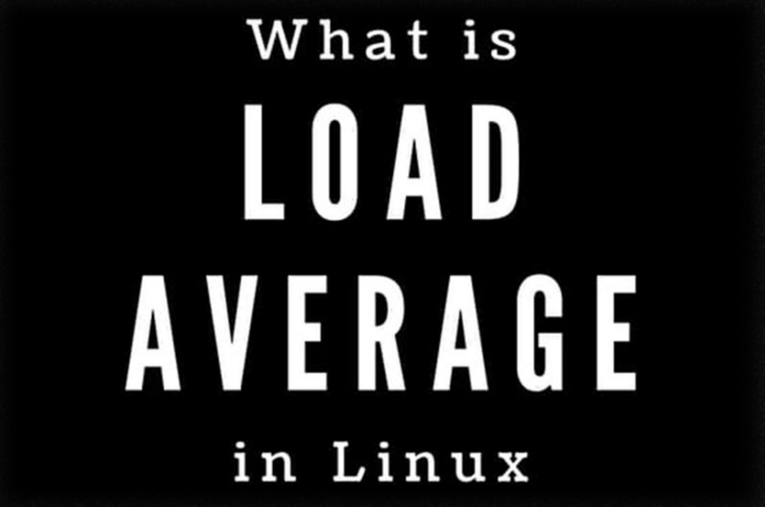
Basics of Load Average in Linux 01

CPU Load Average Calculation 02

The Problem Appear 03

The origin of uninterruptible 04





What is

LOAD AVERAGE

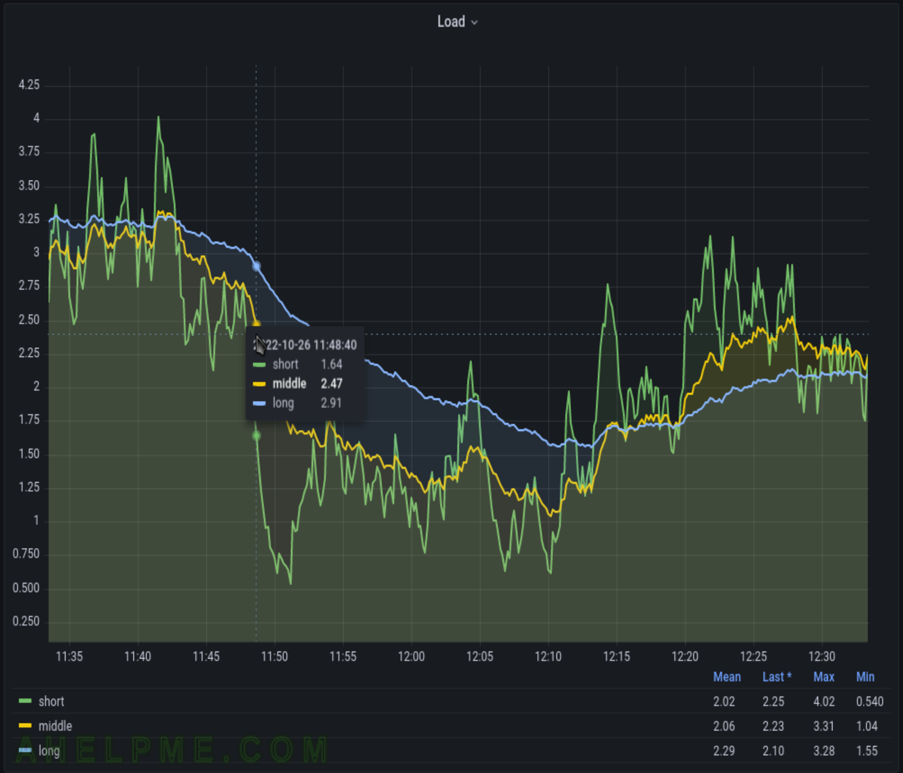
in Linux

Introduction

Load Average in Linux is a metric that is used by Linux users to keep track of system resources. It also helps you monitor how the system resources are engaged.

01

Basics of Load Average in Linux



22-10-26 11:48:40

short	1.64
middle	**2.47**
long	2.91

	Mean	Last *	Max	Min
short	2.02	2.25	4.02	0.540
middle	2.06	2.23	3.31	1.04
long	2.29	2.10	3.28	1.55

Load? Why Load Average?

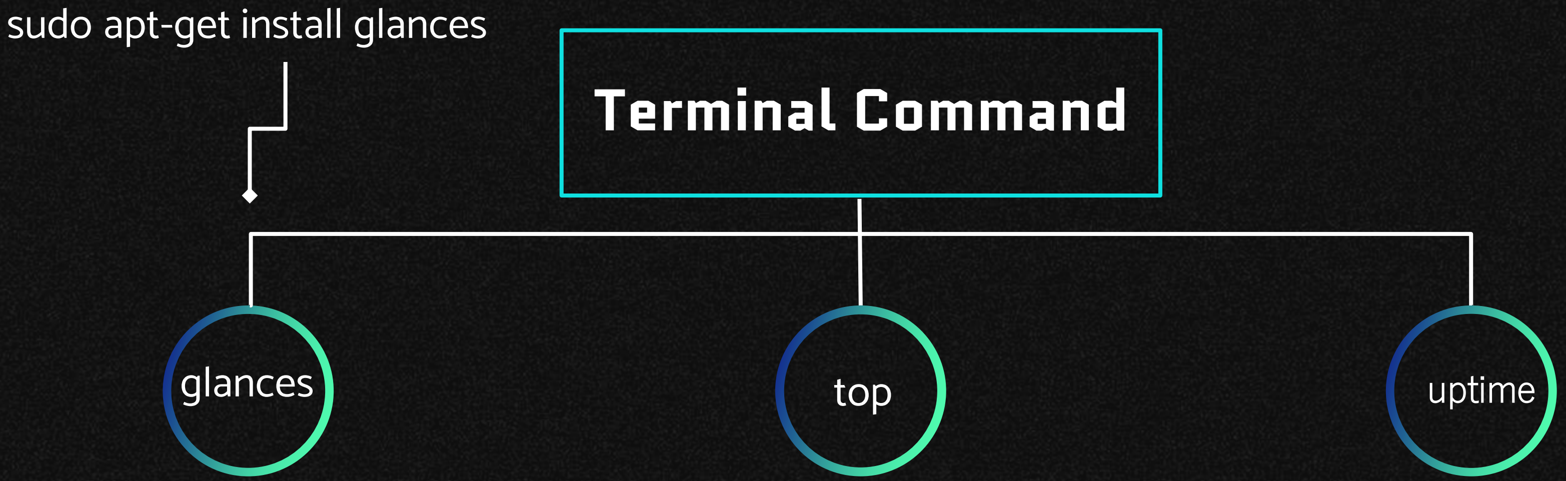
Theory

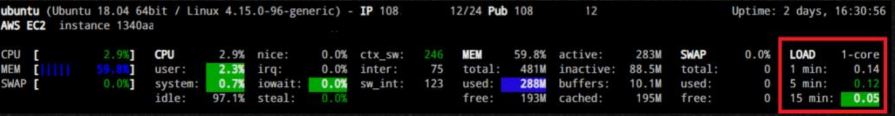
It refers to the number of processes which are either currently being executed by the CPU or are waiting for execution.

An idle system has a load of 0. With each process that is being executed or is on the waitlist, the load increases by one.

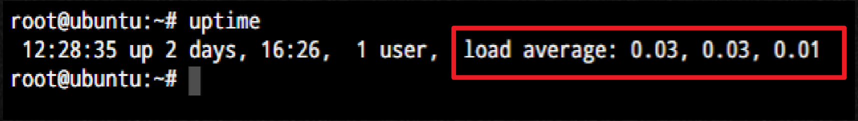
On its own, **the load doesn't give any useful information** to the user. **The load can change in split seconds.** This is because the number of processes using or waiting for the CPU time doesn't remain constant. This is why we use Load Average in Linux to monitor resource usage.

How to Check the Load Average in Linux

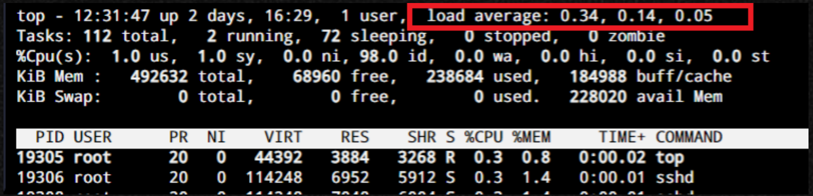




```
ubuntu (Ubuntu 18.04 64bit / Linux 4.15.0-96-generic) - IP 108        12/24 Pub 108        12              Uptime: 2 days, 16:30:56
AWS EC2  instance 1340aa

CPU  [            2.9%]  CPU      2.9%  nice:     0.0%  ctx_sw:  246  MEM     59.8%  active:     283M  SWAP     0.0%  LOAD    1-core
MEM  [|||||      59.8%]  user:    2.3%  irq:      0.0%  inter:    75  total:   481M  inactive:  88.5M  total:      0  1 min:    0.14
SWAP [            0.0%]  system:  0.7%  iowait:   0.0%  sw_int:  123  used:    288M  buffers:   10.1M  used:       0  5 min:    0.12
                         idle:   97.1%  steal:    0.0%                free:    193M  cached:     195M  free:       0  15 min:   0.05
```



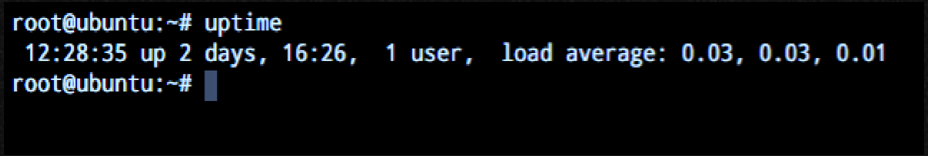
```
root@ubuntu:~# uptime
 12:28:35 up 2 days, 16:26,  1 user,  load average: 0.03, 0.03, 0.01
root@ubuntu:~# 
```



```
top - 12:31:47 up 2 days, 16:29,  1 user,  load average: 0.34, 0.14, 0.05
Tasks: 112 total,   2 running,  72 sleeping,   0 stopped,   0 zombie
%Cpu(s):  1.0 us,  1.0 sy,  0.0 ni, 98.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
KiB Mem :   492632 total,    68960 free,   238684 used,   184988 buff/cache
KiB Swap:        0 total,        0 free,        0 used.   228020 avail Mem

  PID USER      PR  NI    VIRT    RES    SHR S %CPU %MEM     TIME+ COMMAND
19305 root      20   0   44392   3884   3268 R  0.3  0.8   0:00.02 top
19306 root      20   0  114248   6952   5912 S  0.3  1.4   0:00.01 sshd
```

What those three numbers really mean?

- The first value depicts the average load on the CPU for the last minute.
- The second gives us the average load for the last 5-minute interval
- The third value gives us the 15-minute average load

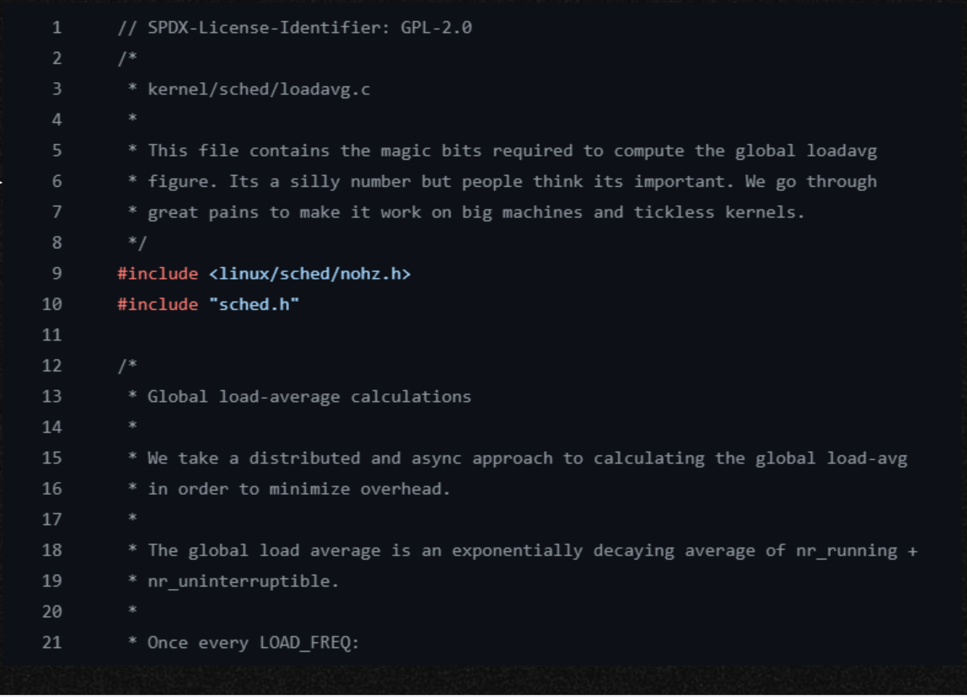


```
root@ubuntu:~# uptime
 12:28:35 up 2 days, 16:26,  1 user,  load average: 0.03, 0.03, 0.01
root@ubuntu:~# 
```

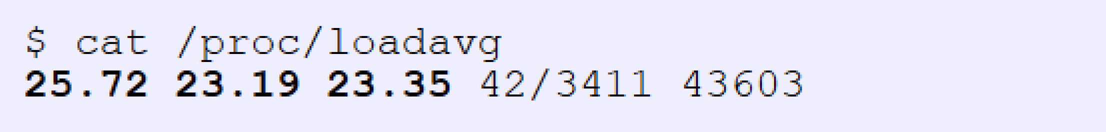
This helps a user get an idea of how the CPU is being utilized by the processes on a system over time.

How to Check the Load Average in Linux

Linus Torvalds
torvalds

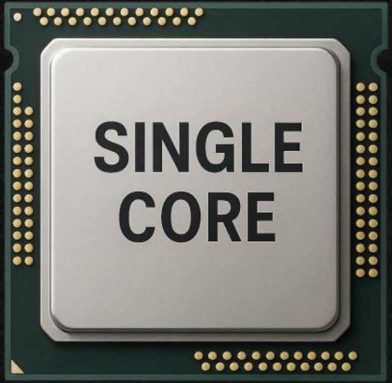


```c
// SPDX-License-Identifier: GPL-2.0
/*
 * kernel/sched/loadavg.c
 *
 * This file contains the magic bits required to compute the global loadavg
 * figure. Its a silly number but people think its important. We go through
 * great pains to make it work on big machines and tickless kernels.
 */
#include <linux/sched/nohz.h>
#include "sched.h"

/*
 * Global load-average calculations
 *
 * We take a distributed and async approach to calculating the global load-avg
 * in order to minimize overhead.
 *
 * The global load average is an exponentially decaying average of nr_running +
 * nr_uninterruptible.
 *
 * Once every LOAD_FREQ:
```

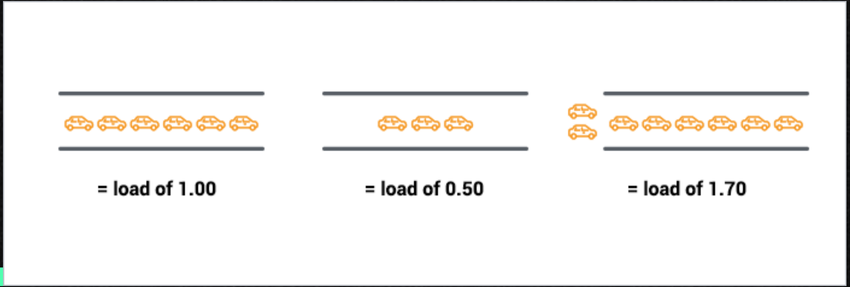


```
$ cat /proc/loadavg
25.72 23.19 23.35 42/3411 43603
```

When should you be worried?



- 0.00 means there's no traffic on the bridge at all.
- 1.00 means the bridge is *exactly* at capacity.
- Over 1.00 means there's backup.



= load of 1.00

= load of 0.50

= load of 1.70

02

CPU Load Average Calculation

F[x]

Δt

$$\text{new\_load} = \text{old\_load} \times e^{-\frac{\Delta t}{\tau}} + \text{current\_load} \times \left(1 - e^{-\frac{\Delta t}{\tau}}\right)$$

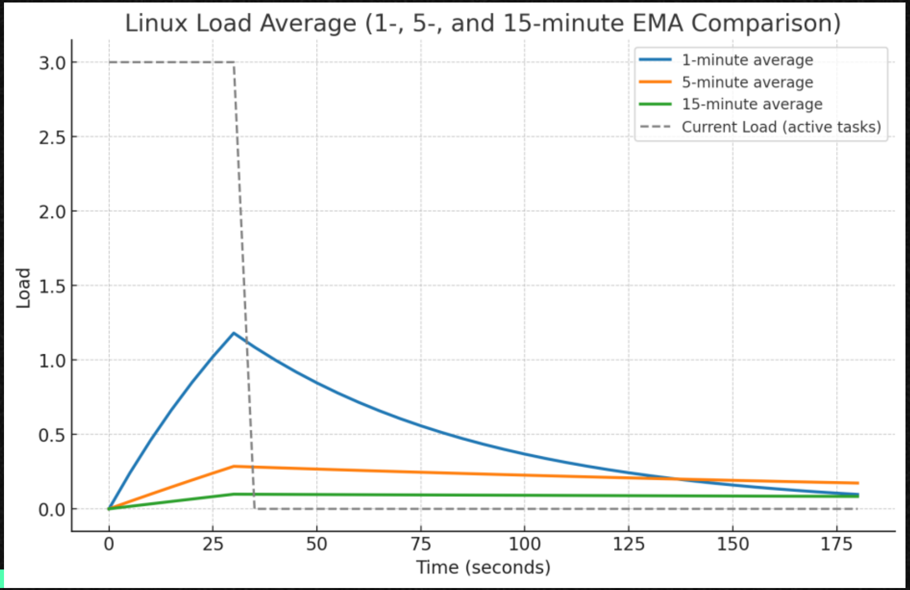
Decay Factor

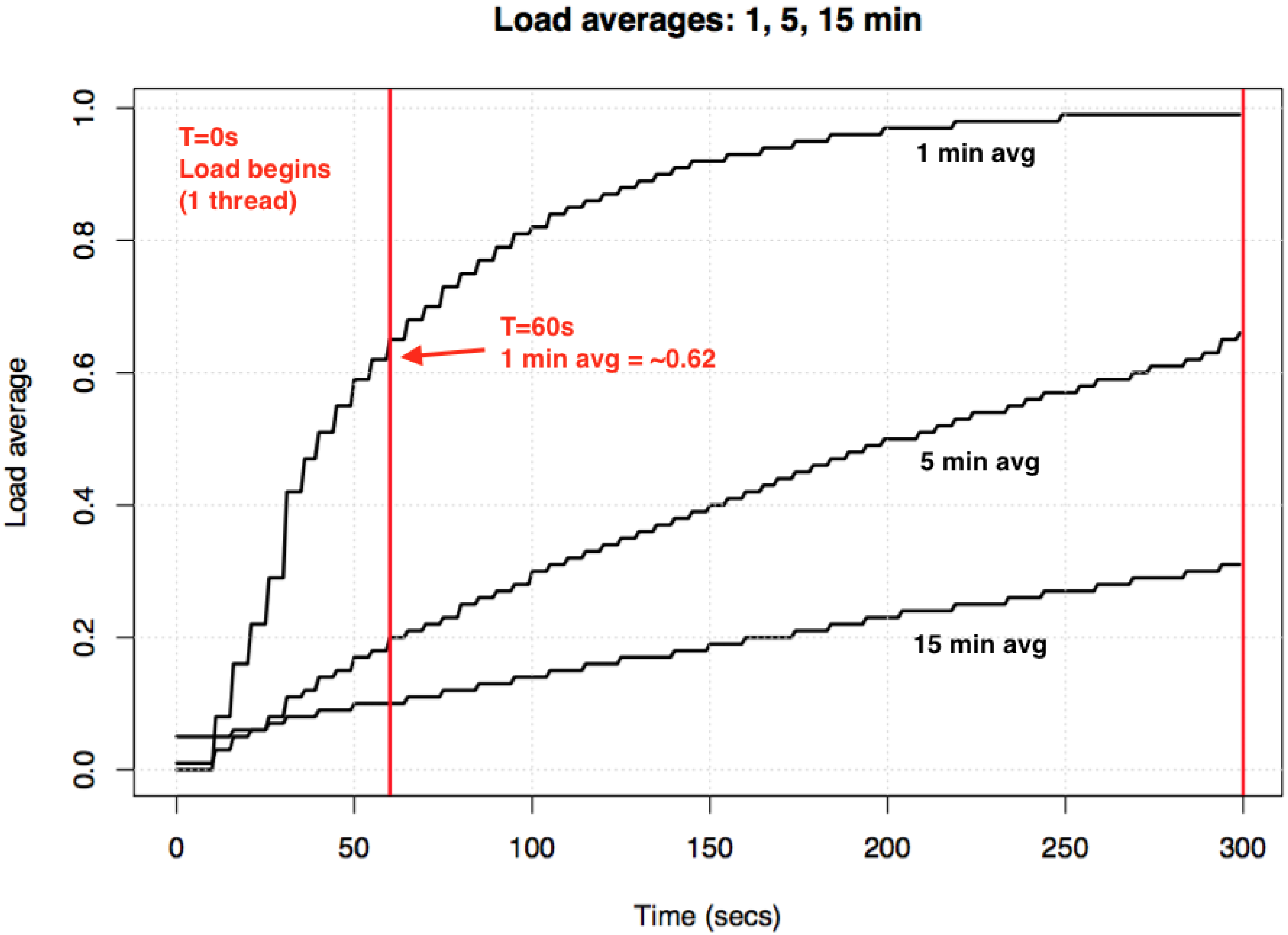
Time Constant

Exponential Moving Average (EMA)

Why Uses *Exponential Moving Average (EMA)?*

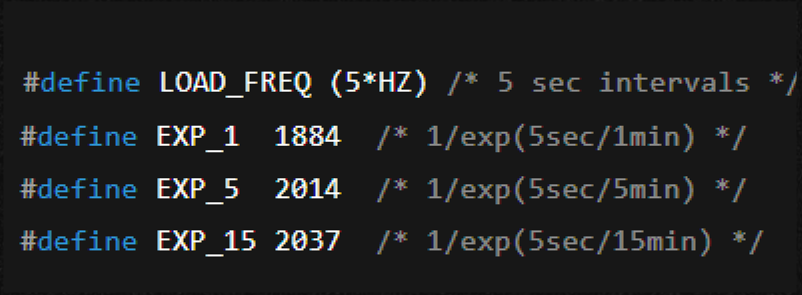
- Prevents rapid fluctuations in displayed load values.
- Provides a better "**sense of system pressure**" instead of raw, noisy data.
- Helps visualize **long-term patterns** of CPU and I/O demand.




Load averages: 1, 5, 15 min
T=0s
Load begins
(1 thread)
T=60s
1 min avg = ~0.62
1 min avg
5 min avg
15 min avg
Load average
0.0
0.2
0.4
0.6
0.8
1.0
0
50
100
150
200
250
300
Time (secs)

Where it comes from?

Inside the Linux kernel source (kernel/sched/loadavg.c



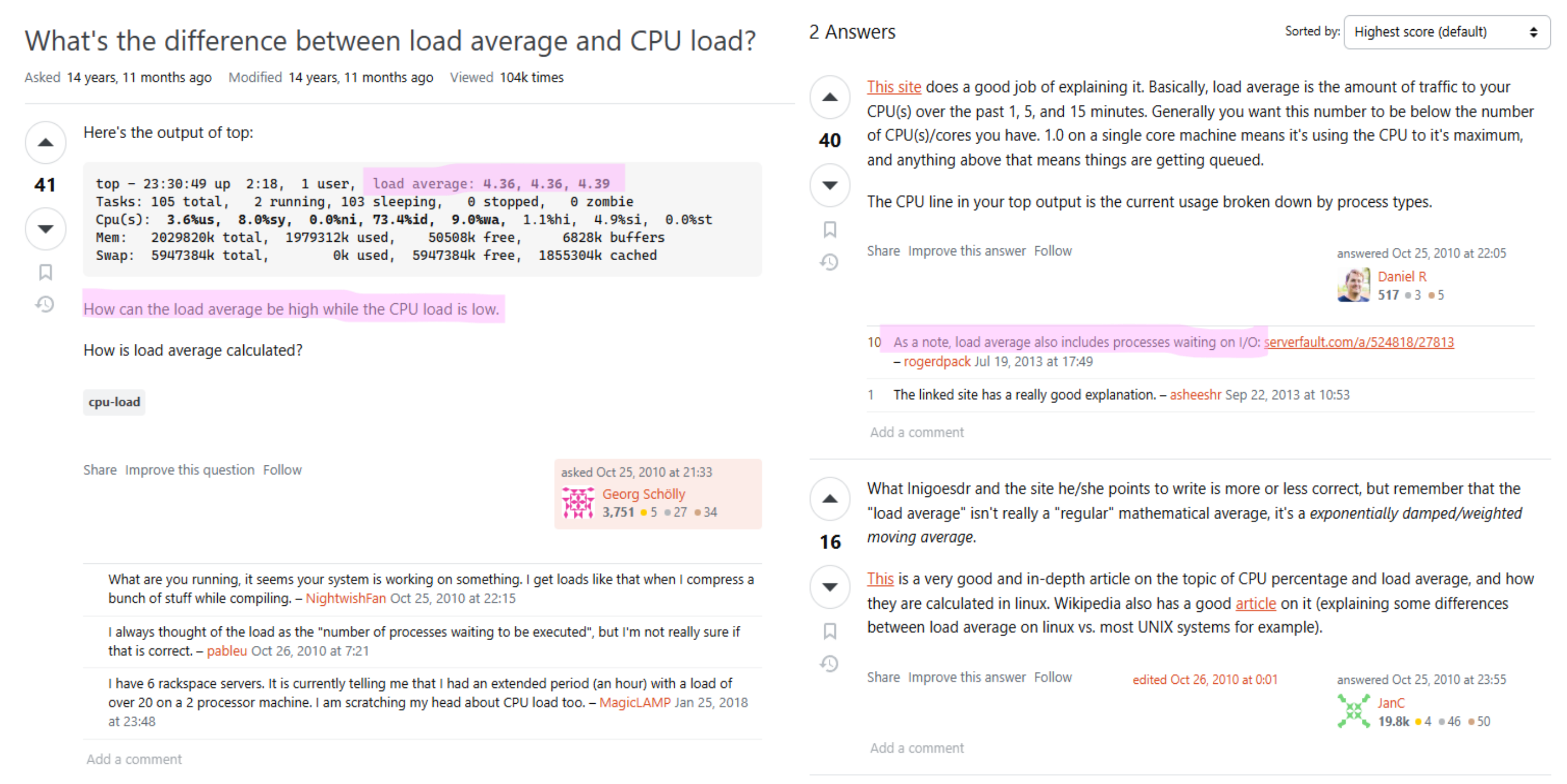
```
#define LOAD_FREQ (5*HZ) /* 5 sec intervals */
#define EXP_1   1884   /* 1/exp(5sec/1min) */
#define EXP_5   2014   /* 1/exp(5sec/5min) */
#define EXP_15 2037   /* 1/exp(5sec/15min) */
```

Average	τ (s)	exp(-5/τ)	In decimal	In kernel scaled form
1 min	60	e^(-5/60)	0.9200	1884
5 min	300	e^(-5/300)	0.9835	2014
15 min	900	e^(-5/900)	0.9945	2037

03

The Problem Appear

The problem appear 😨



What's the difference between load average and CPU load?

Asked 14 years, 11 months ago Modified 14 years, 11 months ago Viewed 104k times

41

Here's the output of top:

```
top - 23:30:49 up  2:18,  1 user,  load average: 4.36, 4.36, 4.39
Tasks: 105 total,   2 running, 103 sleeping,   0 stopped,   0 zombie
Cpu(s):  3.6%us,  8.0%sy,  0.0%ni, 73.4%id,  9.0%wa,  1.1%hi,  4.9%si,  0.0%st
Mem:   2029820k total,  1979312k used,    50508k free,     6828k buffers
Swap:  5947384k total,        0k used,  5947384k free,  1855304k cached
```

How can the load average be high while the CPU load is low.

How is load average calculated?

cpu-load

Share Improve this question Follow

asked Oct 25, 2010 at 21:33
Georg Schölly
3,751 ● 5 ● 27 ● 34

What are you running, it seems your system is working on something. I get loads like that when I compress a bunch of stuff while compiling. – NightwishFan Oct 25, 2010 at 22:15

I always thought of the load as the "number of processes waiting to be executed", but I'm not really sure if that is correct. – pableu Oct 26, 2010 at 7:21

I have 6 rackspace servers. It is currently telling me that I had an extended period (an hour) with a load of over 20 on a 2 processor machine. I am scratching my head about CPU load too. – MagicLAMP Jan 25, 2018 at 23:48

Add a comment

2 Answers

Sorted by: Highest score (default)

40

This site does a good job of explaining it. Basically, load average is the amount of traffic to your CPU(s) over the past 1, 5, and 15 minutes. Generally you want this number to be below the number of CPU(s)/cores you have. 1.0 on a single core machine means it's using the CPU to it's maximum, and anything above that means things are getting queued.

The CPU line in your top output is the current usage broken down by process types.

Share Improve this answer Follow

answered Oct 25, 2010 at 22:05
Daniel R
517 ● 3 ● 5

10 As a note, load average also includes processes waiting on I/O: serverfault.com/a/524818/27813 – rogerdpack Jul 19, 2013 at 17:49

1 The linked site has a really good explanation. – asheeshr Sep 22, 2013 at 10:53

Add a comment

16

What Inigoesdr and the site he/she points to write is more or less correct, but remember that the "load average" isn't really a "regular" mathematical average, it's a *exponentially damped/weighted moving average*.

This is a very good and in-depth article on the topic of CPU percentage and load average, and how they are calculated in linux. Wikipedia also has a good article on it (explaining some differences between load average on linux vs. most UNIX systems for example).

Share Improve this answer Follow

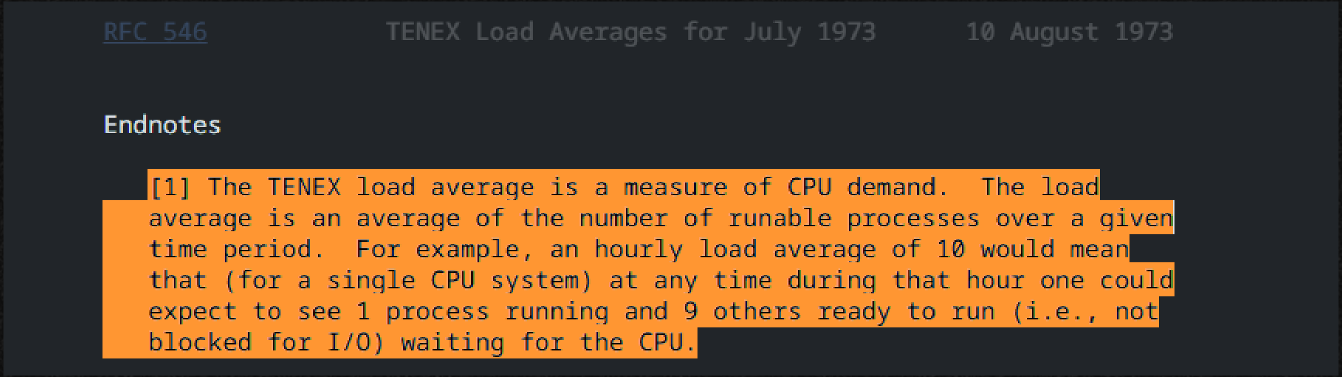
edited Oct 26, 2010 at 0:01

answered Oct 25, 2010 at 23:55
JanC
19.8k ● 4 ● 46 ● 50

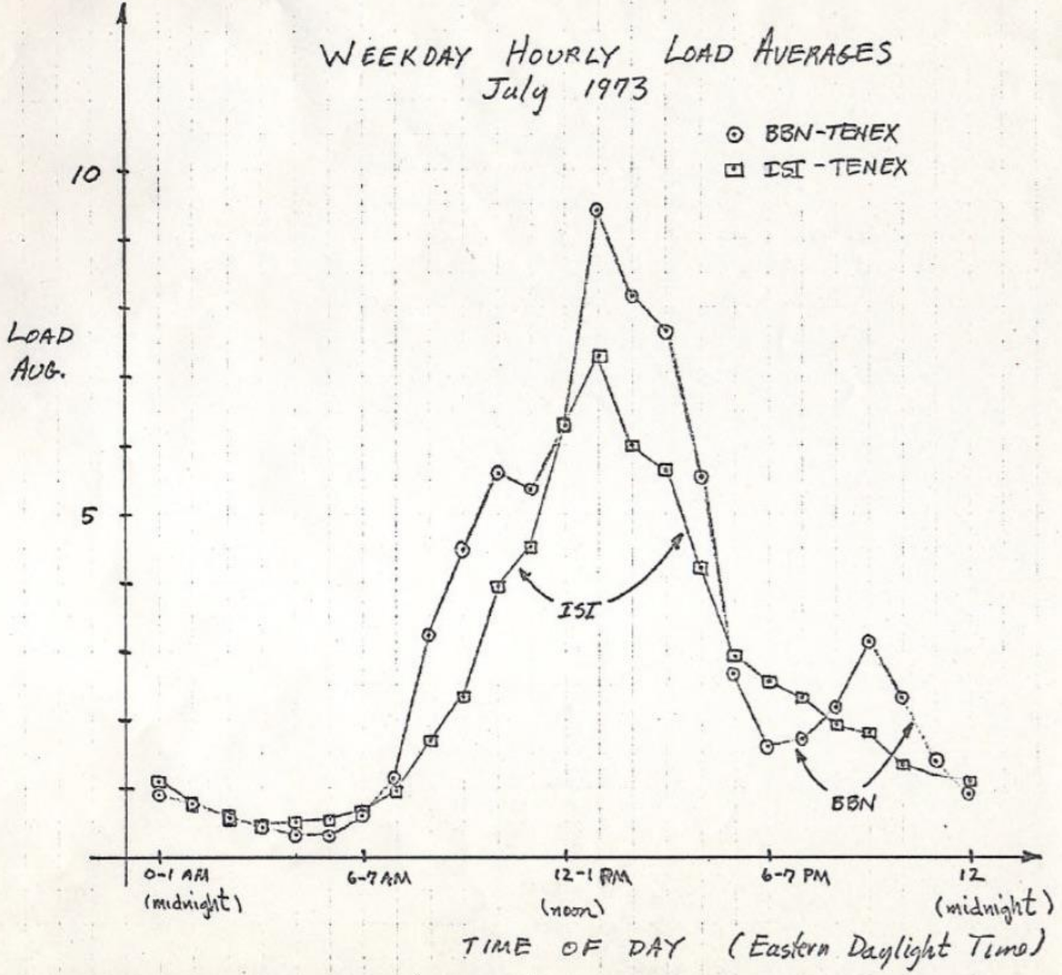
Add a comment

Historty

There's a nice description of this in RFC 546 titled "TENEX Load Averages", August 1973:

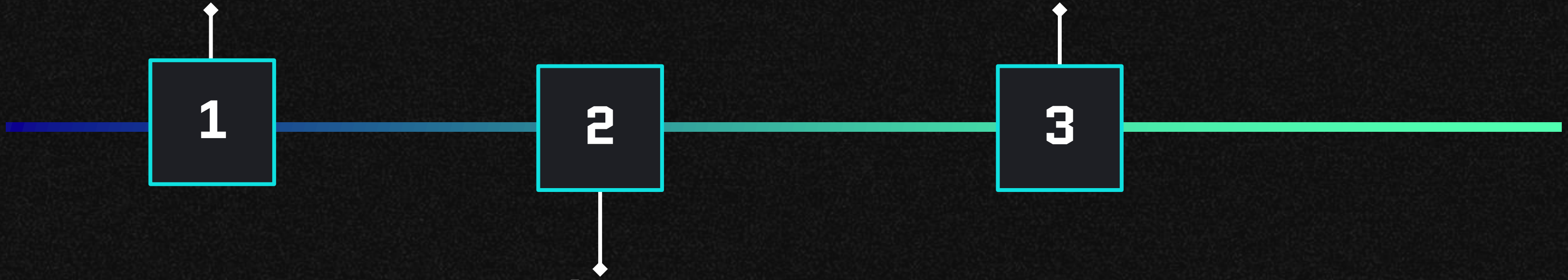


```
RFC 546             TENEX Load Averages for July 1973        10 August 1973


Endnotes

   [1] The TENEX load average is a measure of CPU demand.  The load
   average is an average of the number of runable processes over a given
   time period.  For example, an hourly load average of 10 would mean
   that (for a single CPU system) at any time during that hour one could
   expect to see 1 process running and 9 others ready to run (i.e., not
   blocked for I/O) waiting for the CPU.
```



WEEKDAY HOURLY LOAD AVERAGES
July 1973

TimeLine



1973 TENEX

Process that currently being executed by the CPU or are waiting for execution.

1991-Linux Born

1993

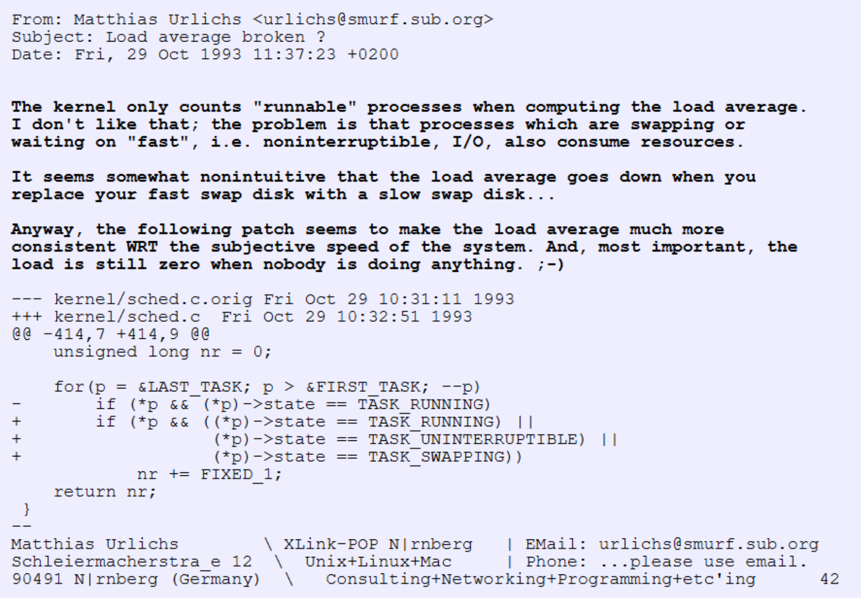
The load averages were deliberately changed to reflect demand for other system resources, not just CPUs. Linux changed from ”CPU load averages” to what one might call ”system load averages”.

04

The origin of uninterruptible

How and where they change it?

Compressed Mailbox File From 1993 On Oldlinux.Org



```
From: Matthias Urlichs <urlichs@smurf.sub.org>
Subject: Load average broken ?
Date: Fri, 29 Oct 1993 11:37:23 +0200


The kernel only counts "runnable" processes when computing the load average.
I don't like that; the problem is that processes which are swapping or
waiting on "fast", i.e. noninterruptible, I/O, also consume resources.

It seems somewhat nonintuitive that the load average goes down when you
replace your fast swap disk with a slow swap disk...

Anyway, the following patch seems to make the load average much more
consistent WRT the subjective speed of the system. And, most important, the
load is still zero when nobody is doing anything. ;-)

--- kernel/sched.c.orig Fri Oct 29 10:31:11 1993
+++ kernel/sched.c  Fri Oct 29 10:32:51 1993
@@ -414,7 +414,9 @@
    unsigned long nr = 0;

    for(p = &LAST_TASK; p > &FIRST_TASK; --p)
-       if (*p && (*p)->state == TASK_RUNNING)
+       if (*p && ((*p)->state == TASK_RUNNING) ||
+                  (*p)->state == TASK_UNINTERRUPTIBLE) ||
+                  (*p)->state == TASK_SWAPPING))
            nr += FIXED_1;
    return nr;
 }
--
Matthias Urlichs        \ XLink-POP N|rnberg   | EMail: urlichs@smurf.sub.org
Schleiermacherstra_e 12  \  Unix+Linux+Mac     | Phone: ...please use email.
90491 N|rnberg (Germany)  \   Consulting+Networking+Programming+etc'ing      42
```

The Linux Process States

- Running or Runnable (R)
- Uninterruptible Sleep (D)
- Interruptable Sleep (S)
- Stopped (T)
- Zombie (Z)

R → running ↔ S/D → T (optional) → Z → gone.

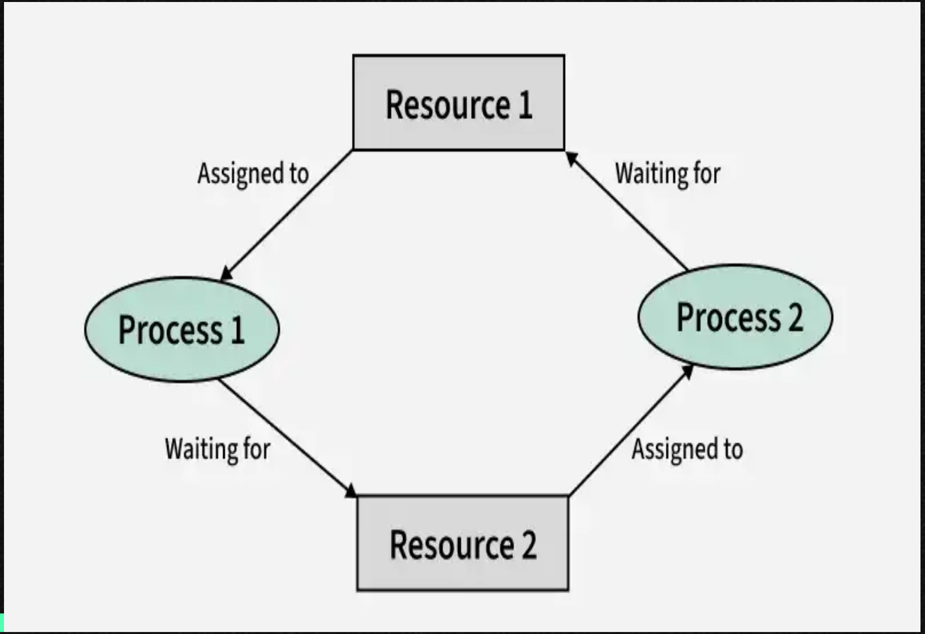
Add Uninterruptible Tasks

When load averages first appeared in Linux, they reflected CPU demand, as with other operating systems. But later on Linux changed them to include not only runnable tasks, but also tasks in the uninterruptible state (TASK_UNINTERRUPTIBLE or nr_uninterruptible).

Adding the uninterruptible state means that Linux load averages can increase due to a disk (or NFS) I/O workload, not just CPU demand.

Causes Of Uninterruptible Tasks (D State)

- Disk I/O (read/write to a block device)
- NFS or network storage operation
- A hardware controller response
- **Kernel lock or driver issue**



Resources

Context:

- ❑ https://www.brendangregg.com/blog/2017-08-08/linux-load-averages.html
- ❑ https://www.digitalocean.com/community/tutorials/load-average-in-linux
- ❑ https://www.scoutapm.com/blog/understanding-load-averages
- ❑ https://superuser.com/questions/1402079/understanding-load-average-and-cpu-in-linux
- ❑ https://github.com/torvalds/linux/blob/master/kernel/sched/loadavg.c
- ❑ https://datatracker.ietf.org/doc/html/rfc546

Photos:

- https://ahelpme.com/software/grafana/create-graph-for-linux-load-average-using-grafana-influxdb-and-collectd/
- https://www.brendangregg.com/blog/images/2017/rfc546.jpg
- https://www.geeksforgeeks.org/operating-systems/introduction-of-deadlock-in-operating-system/
- https://github.com/torvalds