How to delete million of files on busy Linux servers (Work out Argument list too long)

Author : admin

If you try to delete more than **131072** of files on Linux with **rm -f ***, where the files are all stored in the same directory, you will get an error:

/bin/rm: Argument list too long.

I've earlier blogged on deleting multiple files on Linux and FreeBSD and this is not my first time facing this error.
Anyways, as time passed, I've found few other new ways to delete large multitudes of files from a server.

In this article, I will explain shortly few approaches to **delete few million of obsolete files to clean some space on your server**.
Here are *3 methods to use to clean your tons of junk files*.

1. Using Linux find command to wipe out millions of files

*a.) Finding and deleting files using find's **-exec** switch:*

find . -type f -exec rm -fv {} \;

This method works fine but it has 1 downside, file deletion is too slow as for each found file external **rm**

command is invoked.

For half a million of files or more, using this method will take "long". However from a server hard disk stressing point of view it is not so bad as, the files deletion is not putting too much strain on the server hard disk.

*b.) Finding and deleting big number of files with find's **-delete** argument*:

Luckily, there is a better way to delete the files, by using **find**'s command embedded **-delete** argument:

find . -type f -print -delete

*c.) Deleting and printing out deleted files with find's **-print** arg*

If you would like to output on your terminal, what files find is deleting in "real time" add *-print*:

find . -type f -print -delete

To prevent your server hard disk from being stressed and hence save your self from server normal operation *"outages"*, it is good to combine **find** command with **ionice**, e.g.:

ionice -c 3 find . -type f -print -delete

Just note, that *ionice* cannot guarantee find's opeartions will not affect severely hard disk i/o requests. On heavily busy servers with high amounts of *disk i/o writes* still applying the ionice will not prevent the server from being hanged! Be sure to always keep an eye on the server, while deleting the files nomatter with or without *ionice.* if throughout find execution, the server gets lagged in serving its ordinary client requests or whatever, stop the execution of the cmd immediately by killing it from another ssh session or tty (if physically on the server).

2. Using a simple bash loop with *rm* command to delete "tons" of files

An alternative way is to use a bash loop, to print each of the files in the directory and issue **/bin/rm** on each of the loop elements (files) like so:

for i in *; do
rm -f $i;
done

If you'd like to print what you will be deleting add an echo to the loop:

for i in $(echo *); do \
echo "Deleting : $i"; rm -f $i; \

The bash loop, worked like a charm in my case so I really warmly recommend this method, whenever you need to **delete more than 500 000+ files in a directory.**

3. Deleting multiple files with perl

Deleting multiple files with perl is not a bad idea at all.
Here is a perl one liner, to delete all files contained within a directory:

perl -e 'for(){((stat)[9]

As you see the script reads the **$dir** defined directory and loops through it, opening file by file and doing a delete for each of its loop elements.
You should already know **PHP is slow**, so this method is only useful if you have to *delete many thousands of files on a shared hosting server with no (ssh) shell access*.

This *php script* is taken from Steve Kamerman's blog . I would like also to express my big gratitude to Steve for writting such a wonderful post. His post actually become inspiration for this article to become reality.

You can also download the php delete million of files script sample here

To use it rename *delete_millioon_of_files_in_a_dir.php.txt* to *delete_millioon_of_files_in_a_dir.php* and run it through a browser .

Note that you might need to run it multiple times, cause many *shared hosting servers* are configured to *exit a php script which keeps running for too long*.
Alternatively the script can be run through shell with PHP cli:

 php -l delete_millioon_of_files_in_a_dir.php.txt.

5. So What is the "best" way to delete million of files on Linux?

In order to find out which method is quicker in terms of execution time I did a home brew benchmarking on my thinkpad notebook.

a) Creating 509072 of sample files.

Again, I used bash loop to create many thousands of files in order to benchmark.
I didn't wanted to put this load on a productive server and hence I used my own notebook to conduct the **benchmarks**. As my notebook is not a server the *benchmarks might be partially incorrect*, however I believe still .they're pretty good indicator on which deletion method would be better.

hipo@noah:~$ mkdir /tmp/test

hipo@noah:~$ cd /tmp/test;
hiponoah:/tmp/test$ for i in $(seq 1 509072); do echo aaaa >> $i.txt; done

I had to wait few minutes until I have at hand 509072 of files created. Each of the files as you can read is containing the sample "aaaa" string.

b) Calculating the number of files in the directory

Once the command was completed to make sure all the 509072 were existing, I used a find + wc cmd to **calculate the directory contained number of files**:

hipo@noah:/tmp/test$ find . -maxdepth 1 -type f |wc -l
509072

real 0m1.886s
user 0m0.440s
sys 0m1.332s

Its intesrsting, using an **ls** command to calculate the files is less efficient than using *find:*

hipo@noah:/tmp/test$ time ls -1 |wc -l
509072

real 0m3.355s
user 0m2.696s
sys 0m0.528s

*c) benchmarking the different file deleting methods with **time***

- Testing delete speed of **find**

hipo@noah:/tmp/test$ time find . -maxdepth 1 -type f -delete
real 15m40.853s
user 0m0.908s
sys 0m22.357s

You see, using *find* to delete the files is not either too slow nor light quick.

- How fast is perl loop in multitude file deletion ?

hipo@noah:/tmp/test$ time perl -e 'for(){(((stat)[9]