serverfault

Copying a large directory tree locally? cp or rsync?

Asked 14 years, 10 months ago Modified 4 years, 2 months ago Viewed 445k times

340

I have to copy a large directory tree, about 1.8 TB. It's all local. Out of habit I'd use `rsync`, however I wonder if there's much point, and if I should rather use `cp`.

I'm worried about permissions and uid/gid, since they have to be preserved in the copy (I know rsync does this). As well as things like symlinks.

The destination is empty, so I don't have to worry about conditionally updating some files. It's all local disk, so I don't have to worry about ssh or network.

The reason I'd be tempted away from rsync, is because rsync might do more than I need. rsync checksums files. I don't need that, and am concerned that it might take longer than cp.

So what do you reckon, `rsync` or `cp`?

linux **unix** **rsync** **file-transfer** **files**

Share Edit Follow Flag

edited Oct 30, 2014 at 14:15
Carolus
139 1 2 10

asked Jul 20, 2009 at 14:36
Amandasaurus
32.4k 70 197 267

3 If rsync does exactly what you want it to do, if you are quite familiar with its usage for this particular application already, and if it functions quickly enough to suit your taste, then why on earth would you want to switch? – eleven81 Jul 20, 2009 at 14:40

3 Because I'm concerned that rsync will take longer than cp, since rsync does lots of checksumming that cp won't do – Amandasaurus Jul 20, 2009 at 15:31

1 The cpu overhead of the checksum is small compared to the disk/network i/o. Unless the disk are on the same system and the OS can do some clever drive-drive copy in the bus controller. – Martin Beckett Jul 20, 2009 at 17:07

3 Checksumming is done on files that differ at the size and timestamp check. If you're paranoid (like after a power outage during copy) you can force checksumming on all files, but on a local transfer, that's usually slower than starting from scratch. – korkman Oct 8, 2012 at 22:53

17 Maybe he's curious about improving his workflow, and doesn't bury his head in the sand thinking he knows everything. This comment really annoys me. – Martin Konecny Oct 25, 2012 at 1:32

18 Answers

Sorted by: Highest score (default)

268

I would use rsync as it means that if it is interrupted for any reason, then you can restart it easily with very little cost. And being rsync, it can even restart part way through a large file. As others mention, it can exclude files easily. The simplest way to preserve most things is to use the `-a` flag – 'archive.' So:

```
rsync -a source dest
```

Although UID/GID and symlinks are preserved by `-a` (see `-lpgo`), your question implies you might want a *full* copy of the filesystem information; and `-a` doesn't include hard-links, extended attributes, or ACLs (on Linux) or the above *nor* resource forks (on OS X.) Thus, for a robust copy of a filesystem, you'll need to include those flags:

```
rsync -aHAX source dest # Linux
rsync -aHE source dest  # OS X
```

The default cp will start again, though the `-u` flag will "*copy only when the SOURCE file is newer than the destination file or when the destination file is missing*". And the `-a` (archive) flag will be recursive, not recopy files if you have to restart and preserve permissions. So:

```
cp -au source dest
```

Share Edit Follow Flag

edited Jan 1, 2016 at 10:13

ELLIOTTCABLE
103 5

answered Jul 20, 2009 at 14:40

Hamish Downer
9,540 6 39 51

5 The -u flag of cp probably isn't the best solution, as it would not detect a partially copied/corrupted file. The nice thing about rsync is that you can have it md5 sum the files to detect differences. – Chad Huneycutt Jul 20, 2009 at 15:10

13 actually, rsync detects local transfers and enables whole-file copy without checksumming automagically. – korkman Oct 8, 2012 at 22:49

30 and --progress which is really handy! – hookenz Nov 28, 2012 at 3:20

19 -P or --progress shows progress for each file individually. It's useful for copying large files, not for many (thousands) small files as it means lots more output which you can't read. It doesn't show the overal progress of all files combined. – SPRBRN Jul 10, 2013 at 8:56

15 @SPRBRN Since rsync version 3.1.0 it supports `rsync --info=progress2` which does show the overall progress of the transfer (as best it can). Useful to give an approximation during large transfers. – rlf Jul 11, 2017 at 14:33

191

When copying to the local file system I tend to use rsync with the following options:

```
# rsync -avhW --no-compress --progress /src/ /dst/
```

Here's my reasoning:

```
-a is for archive, which preserves ownership, permissions etc.
-v is for verbose, so I can see what's happening (optional)
-h is for human-readable, so the transfer rate and file sizes are easier to read (optional)
-W is for copying whole files only, without delta-xfer algorithm which should reduce CPU load
--no-compress as there's no lack of bandwidth between local devices
--progress so I can see the progress of large files (optional)
```

I've seen 17% faster transfers using the above rsync settings over the following tar command as suggested by another answer:

```
# (cd /src; tar cf - .) | (cd /dst; tar xpf -)
```

Share Edit Follow Flag

edited Dec 9, 2019 at 13:46

answered May 7, 2013 at 19:09

Ellis Percival

2,032 1 12 5

2 I am having following error: `rsync: --no-compress: unknown option` @Ellis Percival. – alper Mar 3, 2018 at 16:31

5 Like @alper, --no-compress wasn't an option for my version of rsync (in CentOS 7); I used --compress-level=0 instead. – Paul Jul 30, 2018 at 23:25

2 above solution `--compress-level=0` on mac – muon Sep 11, 2020 at 1:16

5 I like this option, but progress can be unreadable if running through a large directory tree. Using `--info=progress2` instead of `--progress` kept the progress info as a single updating line. – Matt M May 23, 2022 at 1:39

88

When I have to copy a large amount of data, I usually use a combination of tar and rsync. The first pass is to tar it, something like this:

```
# (cd /src; tar cf - .) | (cd /dst; tar xpf -)
```

Usually with a large amount of files, there will be some that tar can't handle for whatever reason. Or maybe the process will get interrupted, or if it is a filesystem migration, the you might want to do the initial copy before the actual migration step. At any rate, after the initial copy, I do an rsync step to sync it all up:

```
# cd /dst; rsync -avPHSx --delete /src/ .
```

Note that the trailing slash on `/src/` is important.

Share Edit Follow Flag

answered Jul 20, 2009 at 15:15

Chad Huneycutt
2,116 1 16 14

7 +1 I've found tar to generally be faster for large copies than rsync. I like the idea of finishing off with a final rsync, too. – Geoff Fritz Jul 20, 2009 at 16:14

21 That's the beauty of this method. You do not need double the space because you never actually create an intermediate tar file. The tar before the pipe packs the data and streams it to stdout, and the tar after the pipe grabs it from stdin and unpacks it. – Chad Huneycutt May 10, 2012 at 0:45

5 I did a cp -a for a 12gb transfer, and this method for a 42gb transfer. The tar method took about 1/4 the time. – NGaida May 23, 2014 at 17:20

3 No need for subshells: `tar -C /src -c . | tar -C /dst -xp` – Watcom Aug 29, 2016 at 1:19

6 I also put `pv` in the middle to be able to watch the progress, estimating the size of all the data using `df`. I also used `--numeric-owner`, as the source disk was from another system and I didn't want `tar` to mess the owners: `tar -C /old-path --numeric-owner -S -c . | pv -tpeba -s 100G | tar -C /new-path --numeric-owner -S -xp` – Petr Nov 5, 2016 at 16:47

|

17

rsync

Here is the rsync I use, I prefer cp for simple commands, not this.

```
$ rsync -ahSD --ignore-errors --force --delete --stats $SRC/ $DIR/
```

cpio

Here is a way that is even safer, cpio. It's about as fast as tar, maybe a little quicker.

```
$ cd $SRC && find . -mount -depth -print0 2>/dev/null | cpio -0admp $DEST &>/dev/null
```

tar

This is also good, and continues on read-failures.

```
$ tar --ignore-failed-read -C $SRC -cf - . | tar --ignore-failed-read -C $DEST -xf -
```

Note those are all just for local copies.

Share Edit Follow Flag

answered Feb 26, 2012 at 17:06

AskApache Htaccess
401 4 6

2 Why do you use the -S and -D flags for rsync? – miyalys Jul 14, 2015 at 11:37

1 `D` preserves Specials and System files, while `S` "handle sparse files efficiently" not sure why rsync wouldn't just do that by default. – Mark Carpenter Jr Mar 16, 2020 at 14:50

1 In what way is cpio safer than rsync? – Dan Stahlke Jun 4, 2021 at 17:25

1 Why do you use --ignore-errors and --ignore-failed-read? Is it for safety? – Tele Jun 25, 2023 at 13:22

16

This thread was very useful and because there were so many options to achieve the result, I decided to benchmark few of them. I believe my results can be helpful to others have a sense of what worked faster.

To move **532Gb** of data distributed among **1,753,200 files** we had those times:

- `rsync` took 232 minutes
- `tar` took 206 minutes
- `cpio` took 225 minutes
- `rsync + parallel` took 209 minutes

On my case I preferred to use `rsync + parallel`. I hope this information helps more people to decide among these alternatives.

The complete benchmark are published here

Share Edit Follow Flag

answered May 11, 2017 at 19:14

arjones
261 2 4

1 404 page not found – Amedee Van Gasse Mar 6, 2018 at 9:01

2 Thanks @AmedeeVanGasse URL have been fixed a short after you reported :) – arjones Apr 10, 2018 at 1:05

3 Why not benchmarking `cp` ? This is the title of the question! – calandoa Apr 17, 2018 at 13:45

@calandoa I think `cp` is insecure, ie: when it breaks you have to start over, that's way I favor options that can resume, ergo `rsync` is my favorite :) – arjones Apr 19, 2018 at 20:07

1 On a 70GB file system, with 48GB data - block size 1024 - with 385000 files and 427730 directories - `cp` was about 3 minutes faster than `rsync` and `find + cpio`. 18m30s to 22min. – sastorsl May 27, 2020 at 13:04

8

The `rsync` command always computes checksums on every byte it transfers.

The command line option `--checksum` only relates to whether checksums of files are used to determine which files to transfer or not, ie:

> `-c, --checksum` skip based on checksum, not mod-time & size"

The manpage also says this:

> Note that rsync always verifies that each transferred file was correctly reconstructed on the receiving side by checking its whole-file checksum, but that automatic after-the-transfer verification has nothing to do with this option's before-the-

transfer "Does this file need to be updated?" check.

So `rsync` also, always, calculates a checksum of the whole file on the receiving side, even when `-c/ --checksum` option is "off".

Share Edit Follow Flag

edited Apr 2, 2018 at 19:57
Patrick Mevzek
10.3k 7 34 43

answered Nov 28, 2012 at 1:20
John
129 1 1

15 While your post added some interesting information here, the rants, and insults decrease the value of your post. This site is not a forum for un-constructive rants. If you were able to modify the source, have you submitted your modifications as a patch? Have you posted your version on github or something? If you feel so strongly about this, it might be better if you tried to do something a bit more constructive instead of being needlessly insulting. – Zoredache Nov 29, 2012 at 21:31

Yeah, the last paragraph wasn't really necessary. – Sherwin Flight Oct 25, 2015 at 3:57

7

Whatever you prefer. Just don't forget the `-a` switch when you decide to use `cp`.

If you really need an answer: I'd use rsync because it's much more flexible. Need to shutdown before copying is complete? Just ctrl-c and resume as soon as your back. Need to exclude some files? Just use `--exclude-from`. Need to change ownership or permissions? rsync will do that for you.

Share Edit Follow Flag

edited Jul 20, 2009 at 15:52

answered Jul 20, 2009 at 14:40

innaM
1,448 9 10

What does the -p flag do again? – Amandasaurus Jul 20, 2009 at 15:32

1 It will Preserver ownership, timestamps and permissions. – innaM Jul 20, 2009 at 15:34

5 cp -a would be better. – David Pashley Jul 20, 2009 at 15:36

Indeed. Answer changed accordingly. – innaM Jul 20, 2009 at 15:53

`rsync -aPhW --protocol=28` helps speed up those large copies with RSYNC. I always go rsync because the thought of being midway through 90GiB and it breaking scares me away from CP

6

Share Edit Follow Flag

edited Nov 28, 2012 at 9:25

growse
8,060 13 76 117

answered Jul 20, 2009 at 16:24

oneguynick
154 2

2 What is the value of using the older protocol in that command string? – ewwhite Nov 28, 2009 at 3:10

1 On a mac machine the older version of Rsync shipped hangs on some newer rsync protocol revs such as 29. Telling it to move to the older protocol makes it NOT check over and over again. – oneguynick Jan 3, 2010 at 5:52

I guess that number 28 is not valid anymore? – SPRBRN Jul 10, 2013 at 8:59

rsync is great, but has issues with really large directory trees because it stores the trees in memory. I was just looking to see if they'd fix this problem when I found this thread.

5

I also found:

http://matthew.mceachen.us/geek/gigasync/

You could also manually break up the tree and run multiple rsyncs.

Share Edit Follow Flag

answered Jul 20, 2009 at 16:14

n3bulous
51 2

14 If you use version 3 it doesn't keep the whole tree in memory if it is big, it uses a incremental-recursion algorithm: samba.org/ftp/rsync/src/rsync-3.0.0-NEWS – Kyle Brandt Jul 20, 2009 at 17:09

4

There are some speed-ups which can be applied to `rsync` :

Avoid

- `-z` / `--compress` : compression will only load up the CPU as the transfer isn't over a network but over RAM.
- `--append-verify` : resume an interrupted transfer. This sounds like a good idea, but it has the dangerous failure case: any destination file the same size (or greater) than the source will be IGNORED. Also, it checksums the whole file at the end, meaning no significant speed up over `--no-whole-file` while adding a dangerous failure case.

Use

- `-S` / `--sparse` : turn sequences of nulls into sparse blocks
- `--partial` or `-P` which is `--partial --progress` : save any partially transferred files for future resuming. Note: files won't have a temporary name, so ensure that nothing else is expecting to use the destination until the whole copy has completed.
- `--no-whole-file` so that anything that needs to be resent uses delta transfer. Reading half of a partially transferred file is often much quicker than writing it again.
- `--inplace` to avoid file copy (but only if nothing is reading the destination until the whole transfer completes)

Share Edit Follow Flag

answered Sep 13, 2019 at 7:38

Tom Hale
1,165 1 13 26

4

You definitely want to give rclone a try. This thing is crazy fast :

```
sudo rclone sync /usr /home/fred/temp -P -L --transfers 64
```

```
Transferred:       17.929G / 17.929 GBytes, 100%, 165.692 MBytes/s, ETA 0s
Errors:                75 (retrying may help)
Checks:            691078 / 691078, 100%
Transferred:       345539 / 345539, 100%
Elapsed time:     1m50.8s
```

This is a local copy from and to a LITEONIT LCS-256 (256GB) SSD.

You can add `--ignore-checksum` on the first run to make it even faster.

Share Edit Follow Flag

edited Jul 15, 2019 at 13:41

bjoster
4,967 5 27 33

answered Jul 15, 2019 at 12:50

Frédéric N.
49 1

There are some limitations to rclone, but it does seem to be a relatively easy way to take full advantage of your CPU and I/O resources. I haven't figured out if there's an easy way to tell it to copy multiple directories from one directory to another, but I can always script that if need be. – randomScott Jan 26, 2021 at 2:02

`rclone` cannot make an identical copy of a Linux/Unix file system. It cannot preserve hard links, and the `-L` option you give above makes it follow symbolic links instead of ignoring them, which will send it into an infinite loop if you have a symlink such as `/usr/local -> .` Do not use this to make a local copy of a directory tree. – Ian D. Allen Feb 17, 2021 at 23:24

@IanD.Allen It depends on what you want. If you want to copy symlinks as symlinks you can use `--links, -l` . For example use `rclone sync -P -l --create-empty-src-dirs --no-update-modtime --transfers 256 --checkers 256 --fast-list /source /destination` You're right about hard links though. Documentation: rclone.org/local – Frédéric N. Feb 19, 2021 at 7:10

rclone does not preserve the permissions and ownership. – Kim Feb 24, 2021 at 2:26

3

When doing local a local directory copy, my experience is that "cp -van src dest" is 20% faster than rsync. As far as restartability, that's what "-n" does. You just need to rm the partially copied file. Not painful unless it's an ISO or some such.

Share Edit Follow Flag

answered Sep 7, 2011 at 7:26

Ron
31 1

`tar` would also do the job, but won't resume from being interrupted like rsync will.

2

Share Edit Follow Flag

answered Jul 20, 2009 at 15:09

pgs
3,581 20 19

An old answer, but isn't TAR for creating compressed archives of files? How could it be used to transfer files like rsync or cp? – Sherwin Flight Oct 25, 2015 at 4:01

@SherwinFlight cd source ; tar cf - . | (cd dest ; tar xf -) – pgs Oct 26, 2015 at 6:12

ARJ IS SO OLD SCHOOL!! I really doubt that ARJ and/or rsync will give performance.

Definitely what I always do is use cpio:

```
find . -print | cpio -pdm /target/folder
```

2

This is almost fast than CP, definitely faster than tar and without pipeing anything.

Share Edit Follow Flag

edited Nov 28, 2012 at 9:25

growse
8,060 13 76 117

answered Sep 9, 2012 at 4:09

Gonzalo Gorosito
29 1

2 "The original cpio and find utilities were written by Dick Haight while working in AT&T's Unix Support Group. They first appeared in 1977 in PWB/UNIX 1.0" - FreeBSD's `cpio` man page. – Chris S Sep 21, 2012 at 2:54

3 `cpio` unfortunately has an 8GB upper limit for files. – user139948 Oct 6, 2012 at 21:24

3 "*without pipeing anything*" [sic]. Except the `find` command, as you listed it, has a pipe in it: `find . -print | cpio -pdm /target/folder` – warren Oct 7, 2015 at 17:15

1

For anyone in need to copy large amount of small files between two local mounts (in my case it was two NFS mounts of a NAS service from a cloud provider):

`cp` was painfully slow. When watching network throughput, I saw it could only saturate about 1 mbps of bandwidth. Then I tried with tar:

```
tar -pc /mnt/old-nas | tar -xpf - -C /mnt/new-nas
```

which could saturate the line fully, between 250-300 mbps.

Tar seems to perform much better when copying between two mountpoints with high latency.

Share Edit Follow Flag

answered Mar 9, 2020 at 1:36

Lukas Helebrandt
11 1

1

If both storages are local, **cp** should transfer data near maximum possible speed. It is not necessary to use a synchronizer if the target directory is empty, but it brings benefits like restartability, possibility to exclude certain files etc.

rsync is strong in copying over network (delta transfer of big files). But **rsync** keeps its internal data in memory, which may cause problems with huge directory trees.

If you are interested in another synchronizer, you might have a look at Fitus/Zaloha.sh. It runs **find** on both directories and prepares scripts with **cp** commands. It keeps its internal data in files, not in memory. It is used as follows:

```
$ Zaloha.sh --sourceDir="test_source" --backupDir="test_backup"
```

If you want it to just generate the **cp** script (but not execute it, which would require extensive display and interaction), use the --noExec option.

Your use case presumably does not require generation of restore scripts: use the --noRestore option. Last, if you have the fast **mawk** installed, make use of it via the --mawk option.

Share Edit Follow Flag

answered Mar 9, 2020 at 11:53

Petas
41 1

0

What if you use ARJ?

```
arj a -jm -m1 -r -je filepack /source
```

where `-jm -m1` are compression levels and `-je` makes it an executable. Now you have a encapsulated bash of files.

Then for extraction to the target map

```
filepack -y
```

where the source map will be made (where `-y` is always accept, overwrite, skip etc)

One can then scp ftp the filepack to the target area and execute it, if that is possible.

Share Edit Follow Flag

edited Nov 26, 2012 at 21:56

Scott Pack
14.9k 10 54 83

answered May 18, 2011 at 6:20

herauthon
1

1 Arj? Didn't that die out in the 80's? – Michael Hampton Nov 26, 2012 at 22:02

maybe the early 90's if you believe wikipedia – hookenz Nov 28, 2012 at 3:22

1 It assumes you have the 50% free disk space for the resulting file which is not true when I try to transfer my nas, and second why would I not just do a tar file? – Soren Nov 15, 2020 at 16:51

Both will work just fine.

0

Share Edit Follow Flag

answered Jul 20, 2009 at 14:41

pauska
19.7k 5 58 75