SLAC-TN-21-001

When to use rsync

Chin Fang
Zettar Inc. Mountain View, California, USA

R. "Les" A. Cottrell
SLAC National Accelerator Laboratory
2575 Sand Hill Road, Menlo Park, California, USA

Ezra Kissel
Lawrence Berkeley National Laboratory
Energy Sciences Network
1 Cyclotron Road
Mail stop 59R3101
Berkeley, CA 94720

Abstract -- We have endeavored to show, using a series of data transfer results obtained from two testbeds, when to use the popular data copying tool rsync and related tools. Tests have been conducted in local area network (LAN) and wide area network (WAN) environments. We conclude that for files in a certain size range and network latency ≦ 10 ms round trip time (RTT), rsync is still useful for data moving tasks in the category 4 of the U.S. DOE Technical Report "Data Movement Categories"[1]. For more demanding data movement requirements, tools of different classes are suggested. Sample histograms from two DOE user facilities[2] are provided to further support our conclusions.

Keywords -- data movement; storage throughput; storage benchmark; aggregation approach; scale-out; automated testing; histograms.

SLAC National Accelerator Laboratory, Stanford University, Stanford, CA 94309

This material is based upon work supported by the U.S. Department of Energy, Office of Science, Office of Basic Energy Sciences, under Contract No. DE-AC02-76SF00515

[1] https://www.osti.gov/biblio/1756618
[2] https://www.energy.gov/science/science-innovation/office-science-user-facilities

Introduction

Rsync was co-created by Andrew Tridgell and Paul Mackerras[3] in the early 1990s, partially for Andrew's Ph.D. dissertation[4] and partially for backing up his wife's system[5]. From such a modest beginning, it has gained worldwide popularity and is bundled with all major Linux distributions. It is often regarded as the go-to data mover tool by many enterprise IT professionals. Despite its popularity, the tool's various problems in dealing with numerous small files, very large files, and large network latencies are frequently encountered by users, as a casual search would reveal. There are also some attempts to address these shortcomings via the approach of aggregation, typically using a wrapper to run multiple rsync instances at the same time. A good example is parsyncfp[6].

Nevertheless, there have been no systematic investigations about rsync's proper range of operation. The rsync man page[5] only states, quoted "rsync - a *fast*, versatile, *remote* (and local) file-copying tool", but neither "fast" nor "remote" are precisely defined. The effectiveness of using the aggregation approach via a wrapper is also not clear either. This report intends to address the lack of knowledge in this regard. Specifically, the report intends to answer this question: "**in today's data-centric world, when to use rsync**" (*with or without using a wrapper*)? The answer is based on the results obtained from a series of automated tests transferring files using two test beds. Both LAN and WAN transfer results are analyzed to come up with a proper range of use for rsync-based tools.

Regarding the "data-centric world", analysts forecast that by 2025 data will exponentially grow by 10 times according to Intel DCG EVP Navin Shenoy[7], highlighted by a number of shocking statistics, e.g. only about 1% of the data is utilized and acted upon. This exponential data growth has caused many new data management problems involving data movement. Furthermore, the problem of misusing data movers is becoming more prevalent and is one of the fundamental causes of such poor data utilization (see, for example, the DOE Technical Report "Data Movement Categories"[1]).

This report only investigates the common bulk transfer scenarios. Rsync does not have certain capabilities that are offered by some purposely designed data movers (PDDMs) or storage management tools, e.g. comparing snapshots (aka snapshot diff[8,9]). So, in all tests source data is unchanging during a transfer. Note also that even with the use of an aggregation tool such as parsyncfp, an rsync-based transfer is not amenable to scale-out. In addition, it should be pointed out that rsync was created before the multi-core revolution and the difficult task of taking advantage of multiple cores had not yet started.

---

[3] https://en.wikipedia.org/wiki/Andrew_Tridgell
[4] https://www.samba.org/~tridge/phd_thesis.pdf
[5] https://download.samba.org/pub/rsync/rsync.1
[6] https://github.com/hjmangalam/parsyncfp
[7] https://newsroom.intel.com/editorials/data-centric-innovation-summit/
[8] https://docs.oracle.com/cd/E36784_01/html/E36835/gkkqz.html
[9] https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/7/html/storage_administration_guide/ch-snapper-status-command#snapper-diff

Test environments

Two test environments were employed: Zettar Inc.'s testbed[10] with a maximum Ethernet bandwidth of 50Gbps (2x25Gbps), and the U.S. DOE Energy Sciences Network (ESnet)'s 100G SDN testbed[11]. Two network latency values are considered:

1. **LAN**: ≤ 1ms RTT
2. **WAN**: ≅ 90 ms RTT

If the RTT between a reader's source and destination differs from the above, we strongly urge you to carry out your own tests. Rsync is not a high-performance data mover. It takes a long time to carry out most transfer tests in this investigation. The two employed testbeds must be used for other projects too.

Test methodology

First we define our use of the term "hyperscale" dataset: it is a dataset which has ≧ 1 million files, or overall size ≧ 1TB, or both. Next, the elements of the methodology are a subset of what has been used for a recent ESnet data mover evaluation[11]. It is standardized not only to reflect the reality but also to be completely fair.

1. (Optional) if representative and simulated datasets for production data are available, the storage benchmark is performed with such datasets, typically sized in ~ 2TB/dataset. They are used for transfer tests as well.
2. (Mandatory) The storage sweep graph is first produced for the employed storage for both the source and target sides.
3. Only hyperscale datasets are used for testing. ESnet data mover evaluation always uses datasets with file size ranging from 1KiB to 1TiB. Each set contains files of the same size. For example, 4x256GiB means the set consists of four 256GiB sized files (*non-compressible*) thus total size is 1TiB. Owing to the low transfer performance of rsync, only selected sets are used for this investigation.
4. The following types of tests are conducted:
    a. rsync at both ends
    b. rsync at the source, rsync daemon (i.e. rsyncd) at the target location
    c. an rsync perl wrapper, parsyncfp, is used at the source end
5. All tests are carried out automatically. The pair of tester scripts, implemented in the bash shell command language, is available in a github repo[12].
6. With the ESnet testbed, both the source and target nodes are Docker containers[11].

Since the most important foundation of any data movement is the attainable storage throughput in the node that runs the data mover[13], thus a storage sweep was conducted in both environments.

---

[10] https://www.youtube.com/watch?v=5qTpGg57p_o
[11] https://www.es.net/assets/Uploads/zettar-zx-dtn-report.pdf
[12] https://github.com/fangchin/test_rsync
[13] https://youtube.com/watch?v=f5C2b7aYlnk

Storage throughput

Notice that for various file sizes, the storage throughput obtained from the Zettar testbed is 30 - 40% lower than the corresponding levels obtained from the ESnet testbed. This is mainly attributed to the fact that Zettar testbed nodes import NVMe SSDs via NVMeoF over FDR (56Gbps) Infiniband. ESnet testbed nodes have in host NVMe SSDs. The results are obtained using the freeware elbencho storage sweep tools[14]. See **FIG**s 1 and 2 below. They carry important implications for presented results.

A precaution and a postulation

Some readers may rush to the conclusion that based on the results shown subsequently, rsync based tools are fine to use for "almost everything", we would like to bring the following to the readers' attention:

1. Both the Zettar testbed and the ESnet 100G SDN testbed are expertly designed and maintained, highly flexible test environments. All tests have also been conducted under controlled conditions, e.g. without any other running applications competing for the available resources.
2. Normally, rsync is unlikely to be used in such "quiescent" environments.
3. Furthermore, at a reader's site at the time of writing, the attainable storage throughput, computing power, and network bandwidth are highly unlikely to be at the same level as the Zettar and ESnet testbeds.

As such, even if the same testing approach and tester scripts are used in the reader's own environment, *it's anticipated that most readers would obtain much lower performance levels*. Next, the following factors are defined:

1. **`Fs`** = (attainable throughput level at the reader's site / ESnet testbed storage throughput)
2. **`Fc`** = (CPU computational power at the reader's site / ESnet testbed computational power)
3. **`El`** = the transfer rate level obtained on the ESnet testbed
4. **`Ub`** = the upper bound of the obtained level at each file size at the reader's site

Assuming a reader follows the same testing methodology and uses the same tester scripts, the following is a postulation:

```
Ub = Fs * Fc * El
```

The postulation is formulated based on the comparison of Zettar testbed and ESnet 100G SDN testbed LAN parsyncfp results. Also see **Appendix A.1**.

Storage sweeps

Presented in this section are two automatically generated storage sweep graphs. **A caution first**: A storage sweep is "a kind of" storage benchmarking. The goal of any storage benchmarking is to "estimate" the attainable throughput for an application running in a host. It is neither to yield a single

[14] https://github.com/breuner/elbencho/tree/master/contrib/storage_sweep

metric for all possible applications, nor to obtain a highest possible number. As such, a storage sweep normally should be configured to simulate the behaviors (e.g. *IO patterns and storage interactions*) of the target application. But since we also evaluate parsyncfp - which much complicates the matter, thus the sweep on each host is conducted using defaults strictly for comparison purposes.

Zettar testbed

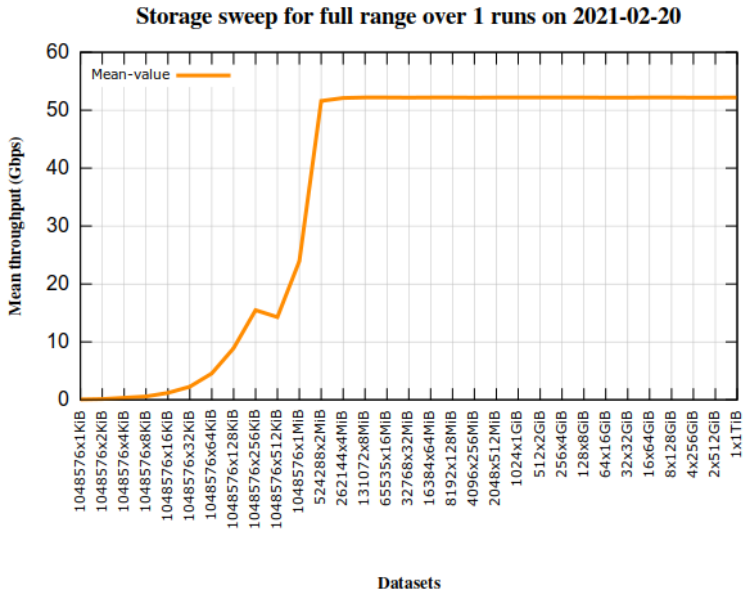

FIG. 1 The elbencho storage sweep graph obtained on a Zettar testbed node

ESnet 100G SDN testbed

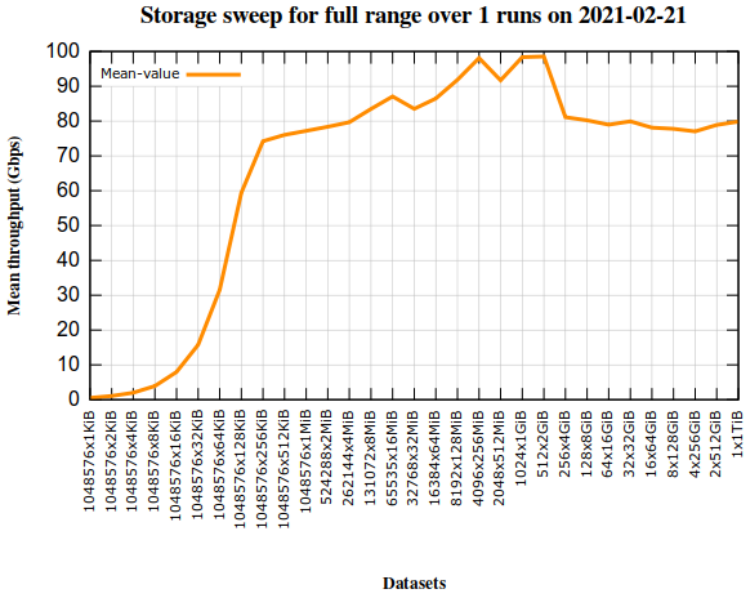

FIG. 2 The elbencho storage sweep graph obtained on an ESnet 100G SDN testbed node

LAN results

Zettar testbed & ESnet 100G SDN testbed

Rsync only and rsync + rsyncd

Despite the fact that the use of rsync + rsyncd is regarded as insecure[15] and not the default way of using rsync, it could be more efficient and achieve faster data transfer rates than rsync (over ssh) only. For completeness we thus also compare rsync and rsync + rsyncd for this investigation.

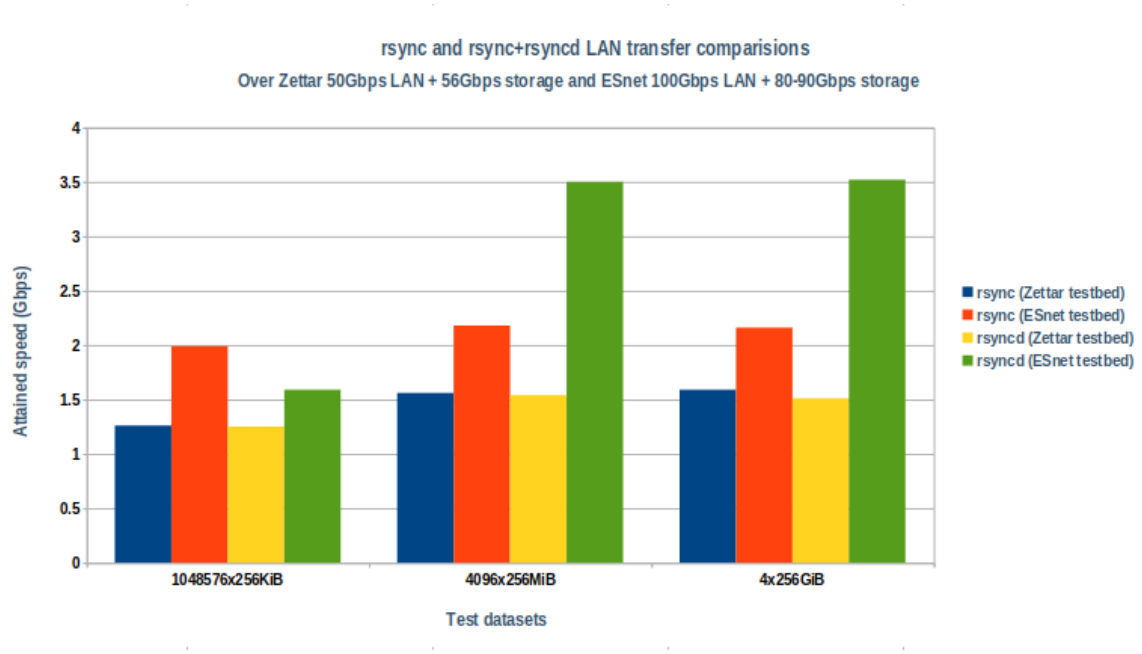

FIG. 3 The rsync only and rsync + rsyncd results were obtained on both the Zettar testbed and the ESnet 100G SDN testbed.

It should be evident that for rsync, the higher attainable storage throughput of the ESnet testbed nodes makes the most significant difference. The use of the rsync daemon (i.e. rsyncd) didn't bring any visible benefits on the Zettar testbed. But the higher attainable storage throughput + more powerful CPUs of ESnet testbed nodes, together with the fact that rsyncd doesn't use the ssh protocol[16] (*note that the default port for rsyncd is 873, not 22*) makes the use of rsyncd more beneficial in that environment. Both the Zettar and ESnet testbed have far more than enough network bandwidth so that aspect is at best of a secondary importance.

[15] https://download.samba.org/pub/rsync/NEWS#3.2.3 - there is a rsync-ssl helper script since 3.2.3. It is not tested due to lack of time. Interested reader is invited to modify the tester scripts then carry out the testing in the reader's own environment

[16] https://download.samba.org/pub/rsync/rsyncd.conf.5.html

Zettar testbed

Parsyncfp

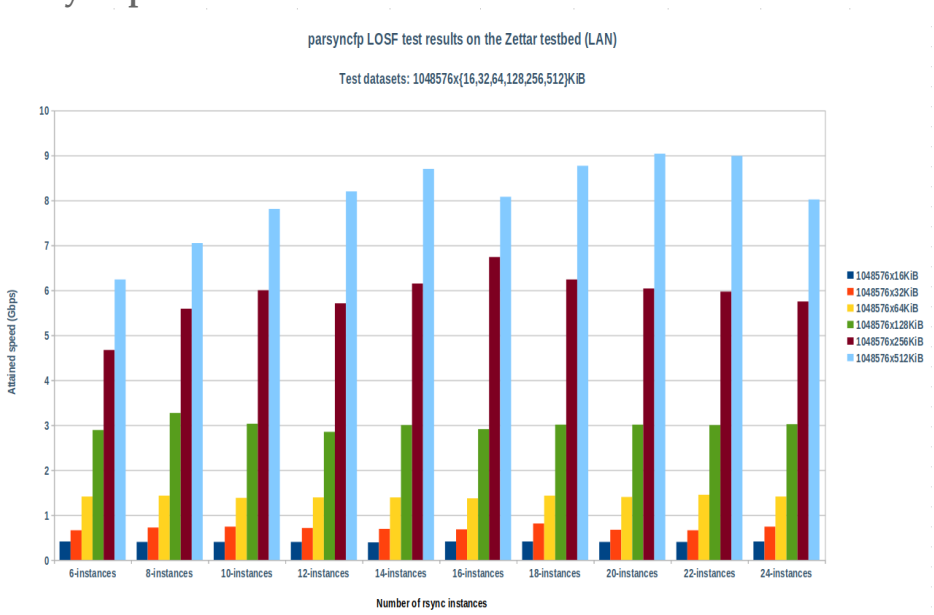

FIG. 4 It is evident even in the LAN, using an aggregated approach with parsyncfp, when file sizes ≤ 128KiB, it's not very fruitful to use rsync as the tool. But for larger file sizes in the LOSF range, it could be useful.

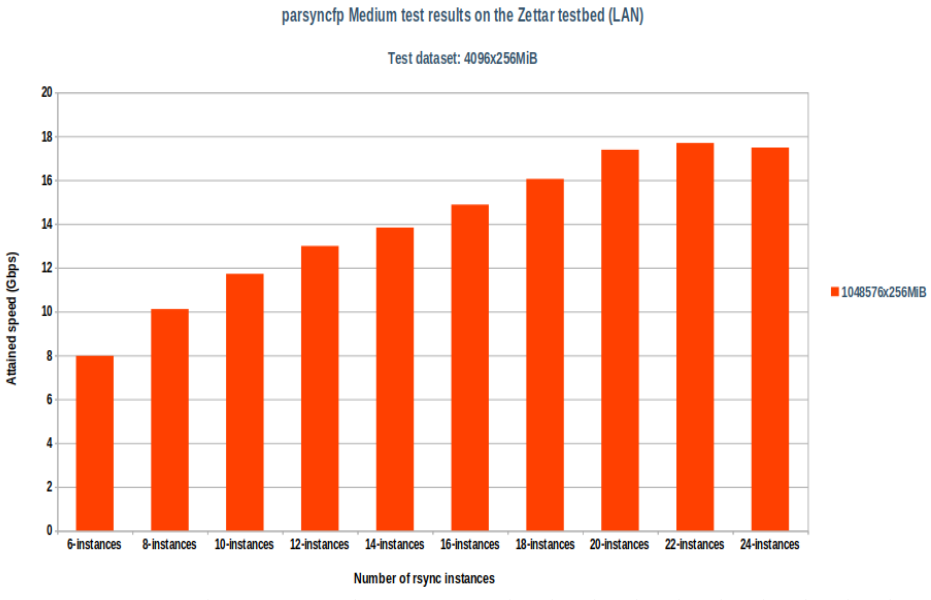

FIG. 5 The results of "sweeping" the number of rsync processes using parsyncfp on the Zettar testbed for a selected dataset with medium-sized file (*4096x256MiB*) in the LAN environment. It is evident that the parsyncfp, with its aggregated approach, could be useful, but this requires the user to be aware of such characteristics of the tool.

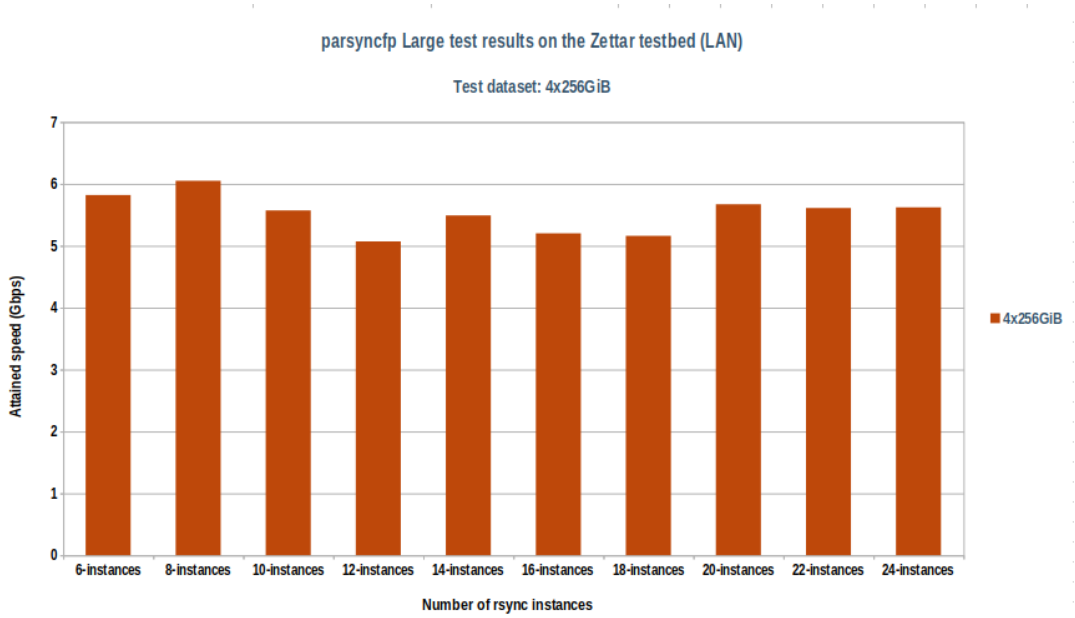

FIG. 6 The results of "sweeping" the number of rsync processes using parsyncfp on the Zettar testbed for a selected dataset with large-sized file (4x256GiB) in the LAN environment. It is clear that even with the parsyncfp, there is no consistent improvement seen with more rsync processes. The intrinsic inefficiency of rsync in dealing with large sized files alluded in the **Introduction** section, shows up in this case.

ESnet 100G SDN testbed

Parsyncfp

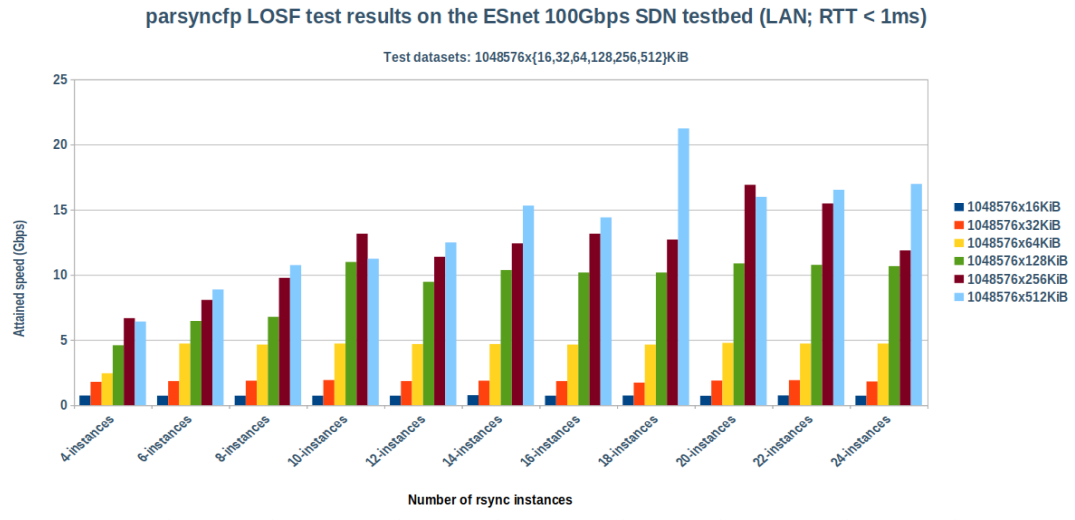

FIG. 8 The results of "sweeping" the number of rsync processes using parsyncfp on the ESnet testbed for LOSF in the LAN environment. The same observations made in the caption of **FIG. 4** apply here as well. Notice that the higher rsync, parsyncfp attained speeds for the ESnet testbed compared to the Zettar testbed are most likely attributed to the higher storage throughput and more powerful CPUs of the ESnet testbed.

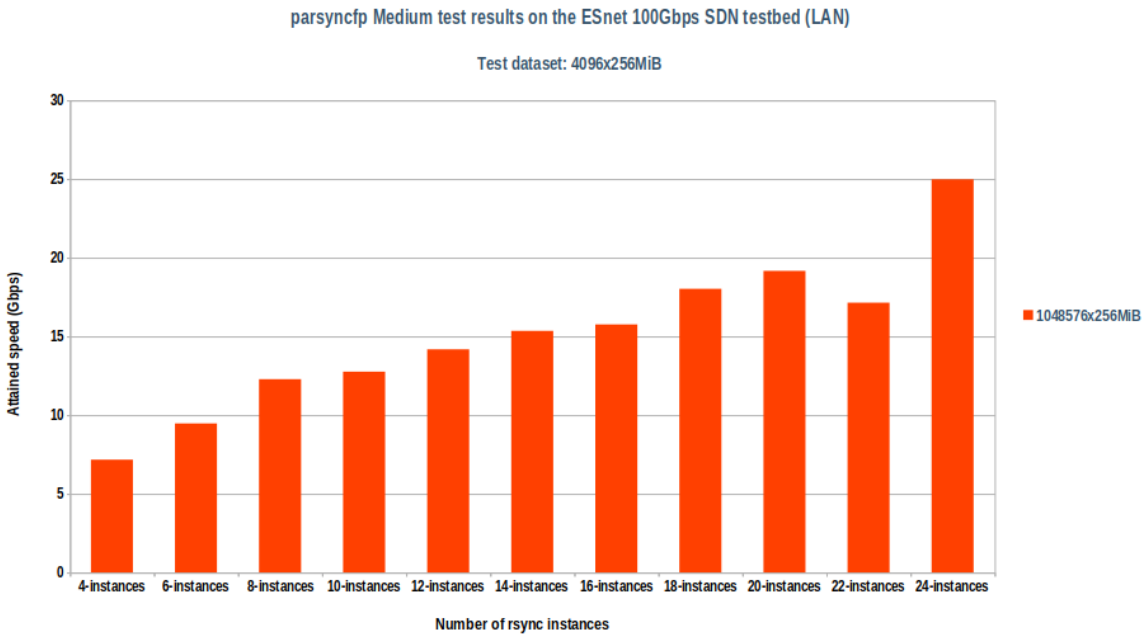

FIG. 9 The results of "sweeping" the number of rsync processes using parsyncfp on the ESnet testbed for a selected dataset with medium-sized file (4096x256MiB) in the LAN environment. The same observations made in the caption for **FIG. 5** apply.

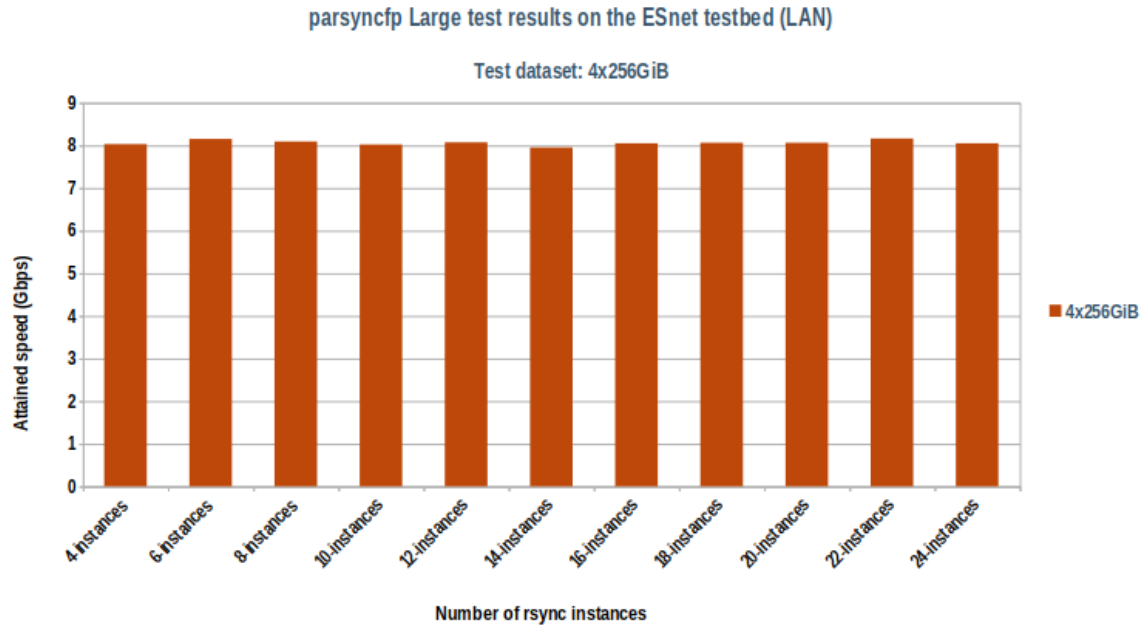

FIG. 10 The results of "sweeping" the number of rsync processes using parsyncfp on the ESnet testbed for a selected dataset with large-sized file (4x256GiB) in the LAN environment. Again, it is clear that even with the parsyncfp, there is no gradual improvement seen with more rsync processes. The intrinsic inefficiency of rsync in dealing with large sized files alluded to in the **Introduction** section, shows up in this case.

WAN results

ESnet 100G SDN testbed

Rsync only and rsync + rsyncd

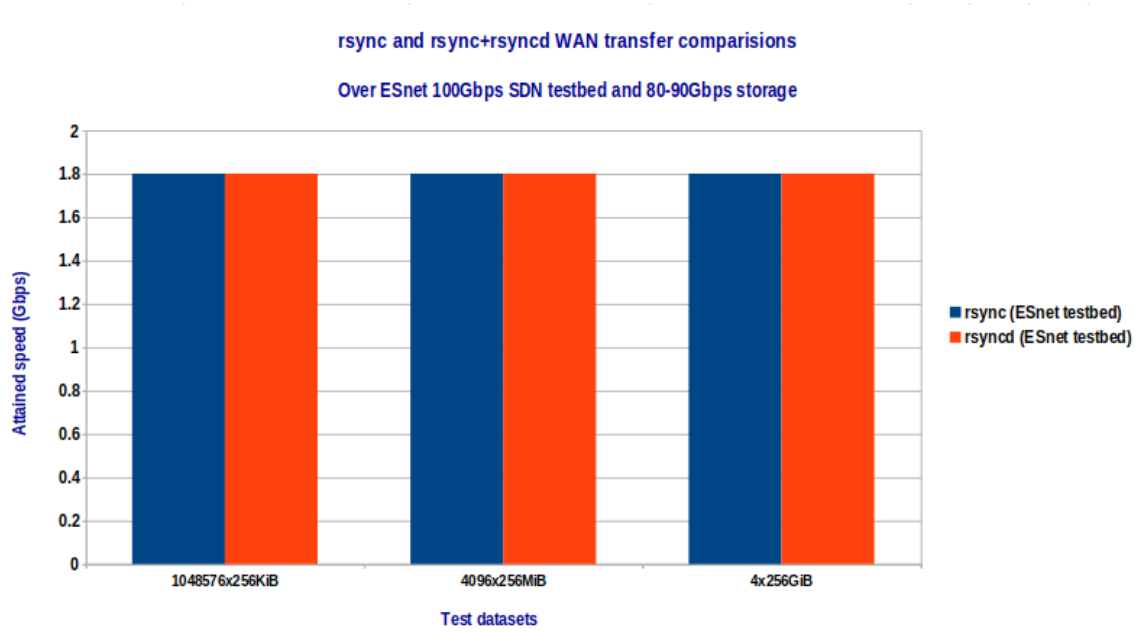

FIG. 11 The rsync only and rsync + rsyncd results obtained on ESnet 100G SDN testbed in the WAN environment (RTT ~ 90ms). It should be evident that neither rsync alone nor both rsync + rsyncd could overcome the tool's inability to handle large network latency – large file sizes didn't help at all. Transferring LOSF over a WAN presents even more challenges to rsync or rsync based tools such as parsyncfp.

Parsyncfp

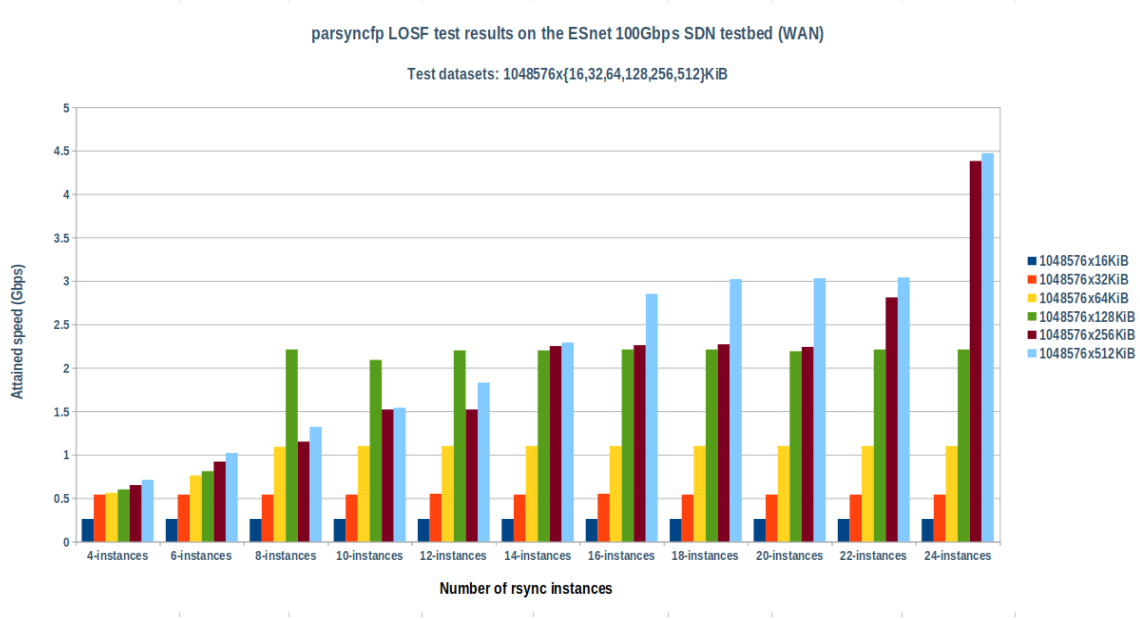

FIG. 12 Contrast to the LAN results, this set of results of "sweeping" the number of rsync processes using parsyncfp on the ESnet testbed for LOSF in the WAN environment should make it clear that despite the abundant storage throughput, computing power available, and generous network bandwidth, plus the use of an aggregated approach, the intrinsic inability of rsync to handle large network latencies simply cannot be masked anymore.

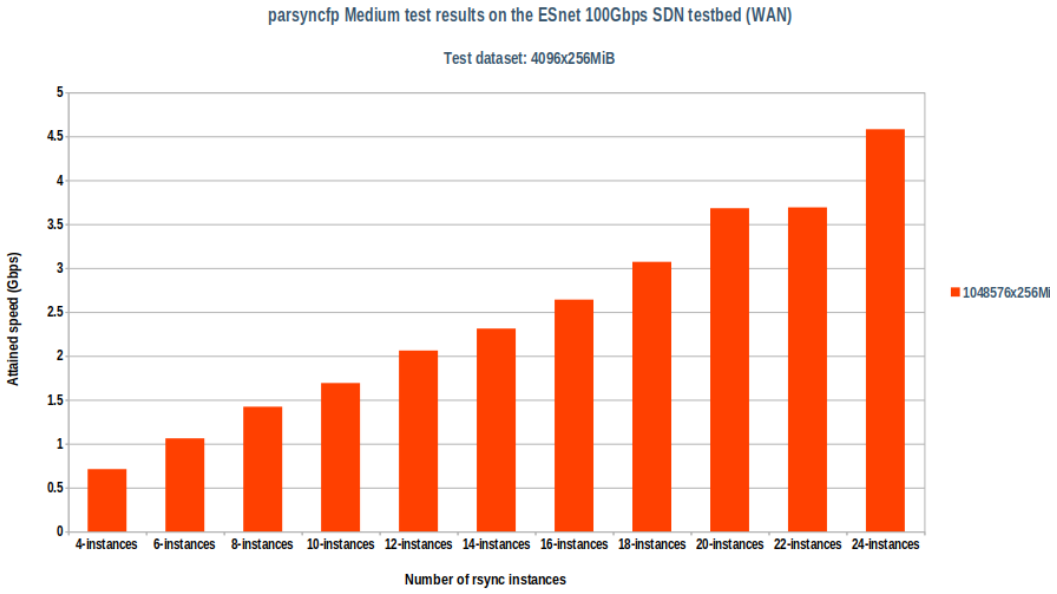

FIG. 13 Owing to the mostly low transfer rates, executing a parsyncfp process number sweep took a long time (***~15 hours***). The outcome shows that as is the case for the LAN based testing, more rsync instances helped. But compared to what PDDMs can do, the result is not that commendable - the highest value is only about 1/4th of the corresponding LAN value.

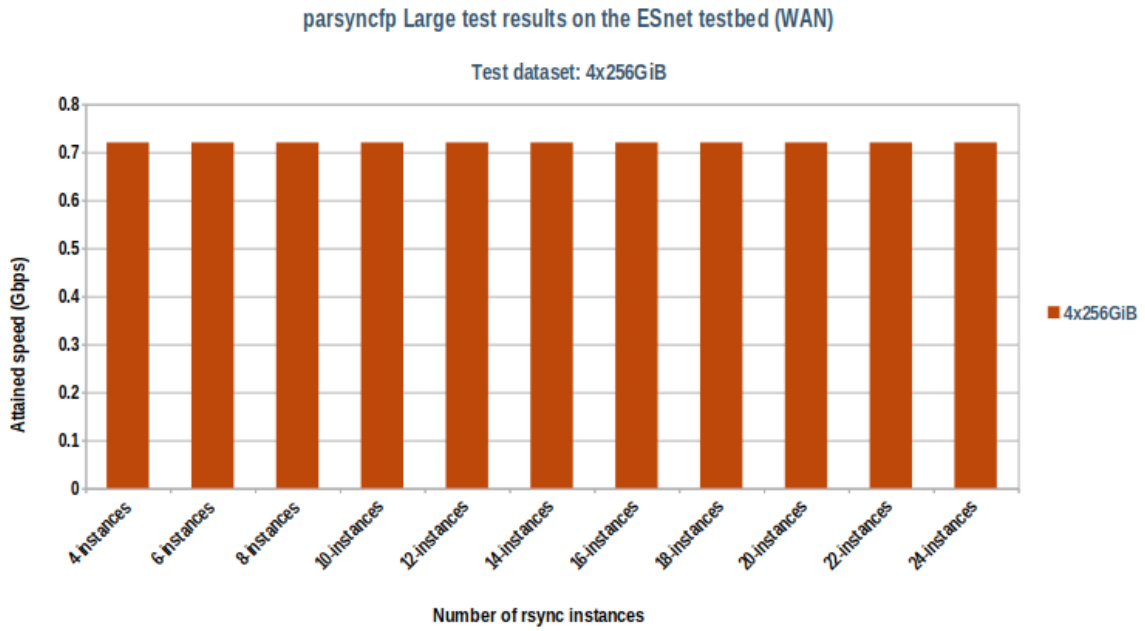

FIG. 14 Due to the really low transfer rate, the entire parsyncfp process number sweep took **~38 hours** to complete. The outcome is nothing commendable.

A glance at two PDDMs

Previously, with many test results, the limitations of rsync based tools are made clear. But then a natural question would be "What are the alternatives"? To answer the question, it suffices to take a glance at two DOE Office of Science[17] funded / invested PDDMs commercial offerings, Globus[18] GridFTP (first funded in 1997) and Zettar zx[19] (invested in 2019). They have published results on ESnet's 100G testbed.[11] It should be evident that

1. PDDMs tend to achieve good to excellent (very close) LAN/WAN results, whereas transfer results of rsync-based tools deteriorate rapidly as network latency increases
2. PDDMs provide overall higher transfer levels than rsync-based tools over the corresponding data file size
3. PDDMs in general provide better LOSF transfer rates than rsync based tools. Likewise for large files. See Reference 11 for an example.

Other factors such as ease of use, scalability, and manageability (both can be automated via scripting) are not even part of rsync's design. Nevertheless, a full and complete treatment comparing and evaluating PDDMs requires another paper.

[17] https://www.energy.gov/science/office-science
[18] https://www.globus.org/subscriptions
[19] https://www.zettar.com/resources/

In passing it should be of interest to note the following:

1. Both Globus GridFTP and Zettar zx, are scale-out capable, i.e. they can run in clusters, and all instances work on the same set of data collaboratively, typically stored in a shared storage (e.g. Lustre[20]).
2. These cluster instances do not need a cluster workload manager (CWM) such as SLURM[21].
3. For scale-out capable data movers, the efficiency of scalability is usually expressed as a percentage: (**`aggregated transfer rates / number of nodes)/single node throughput x 100/100`**. This value is called "linear scalability efficiency". The closer it is to one, the better.

Sample histograms of modern research data

Below are a few histograms collected from the U.S. DOE Joint Genome Institute[22] and Linac Coherent Light Source (LCLS)[23], SLAC National Accelerator Laboratory. From such histograms, it should be clear that many important research datasets are dominated by small files. In addition, AI, machine learning, and deep learning are a few other well-known areas where small files are common. For example, for deep learning, LOSF reads and small random IOPS for large files are prevalent[24]. But note also that some LCLS experiments have output files in the multiple TB range. Thus, using rsync as a data mover in many cases today would be highly suboptimal.

Histograms from JGI

[20] https://www.lustre.org
[21] https://slurm.schedmd.com/overview.html
[22] https://jgi.doe.gov
[23] https://lcls.slac.stanford.edu
[24] https://parallel.computer/presentations/PPoPP2021/elbenchoANewStorageBenchmarkForAIetal.pdf

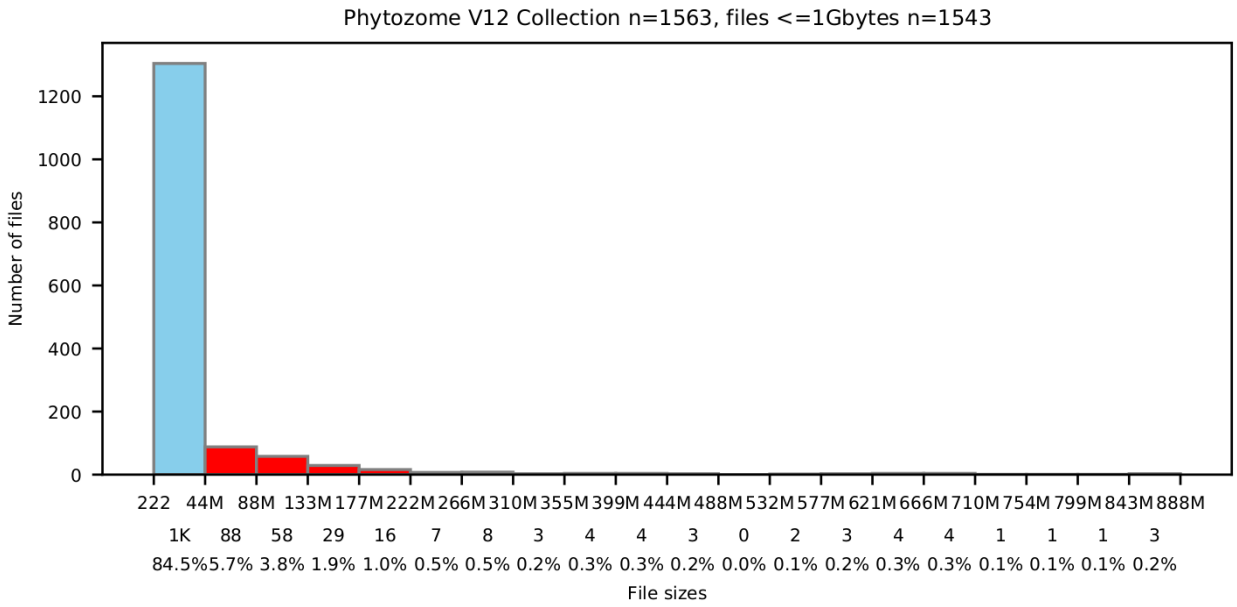

FIG. 15 Of the 1563 files, 1543 files have sizes ≦ 1GB

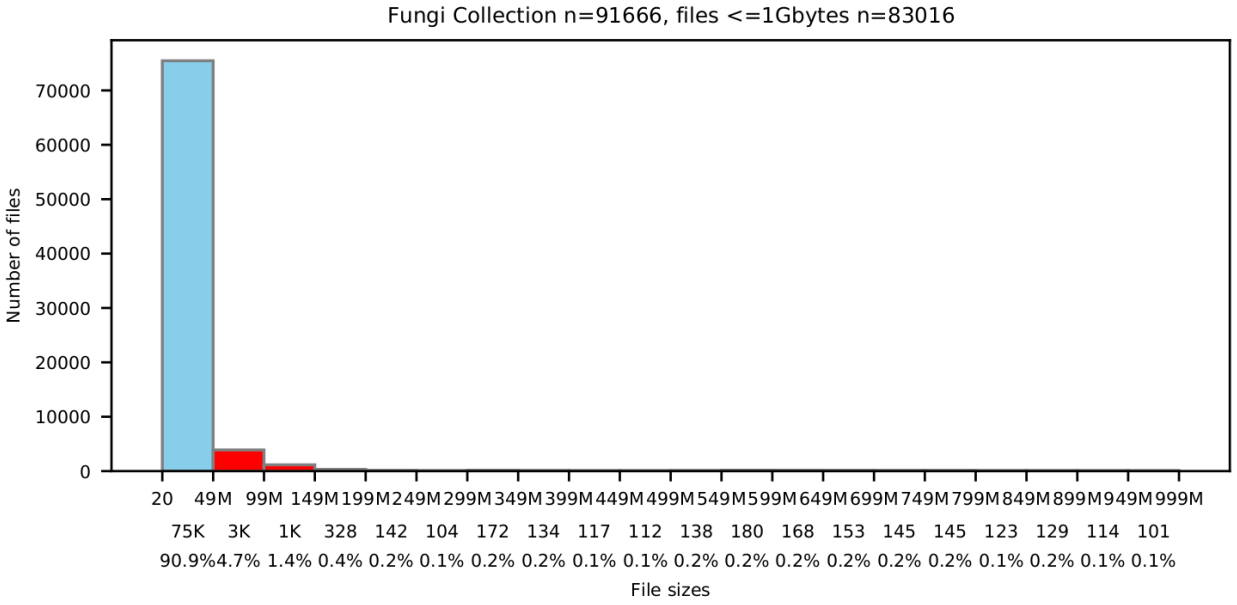

FIG. 16 Of the 91666 files, 83016 files have sizes ≦ 1GB

Histograms from LCLS/SLAC

The first two experiments are typically ones with large detectors but the maximum file size is limited to certain values. The third one has no restrictions on the file size and therefore files can grow to multiple TB in size. The fourth histogram is for an experiment that has a low data rate.

The large number of small files is mostly due to files that record the md5 value of a data file. For each data file there are two index files and two md5 files.

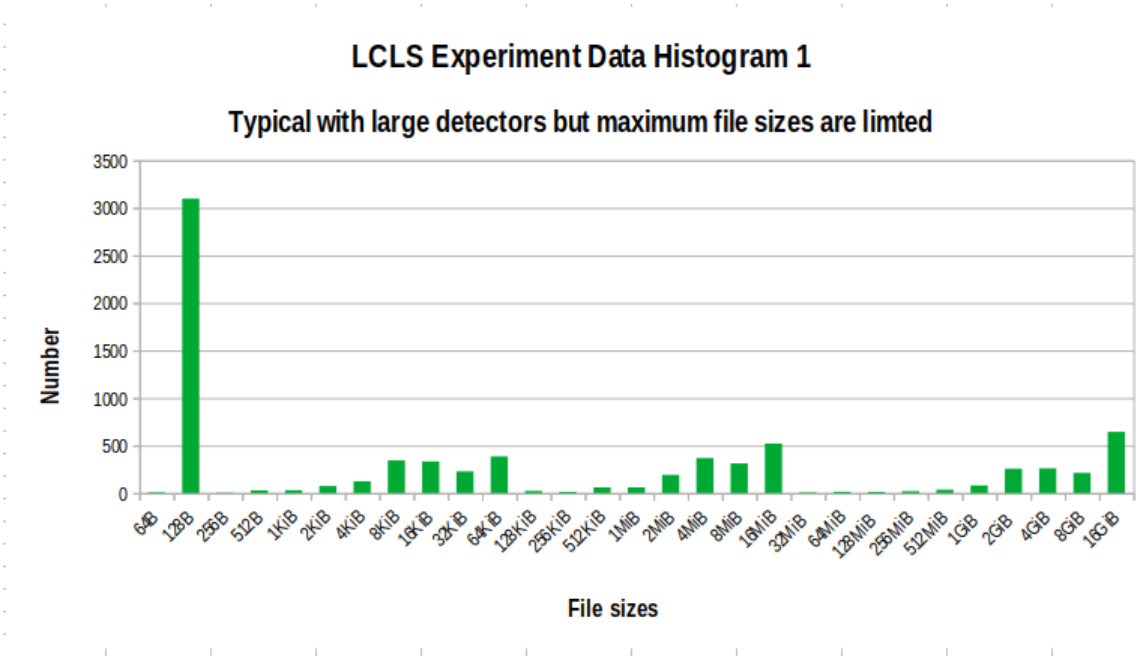

FIG. 17 It is evident for this LCLS experiment, the majority of the files have sizes $\leqq$ 256B

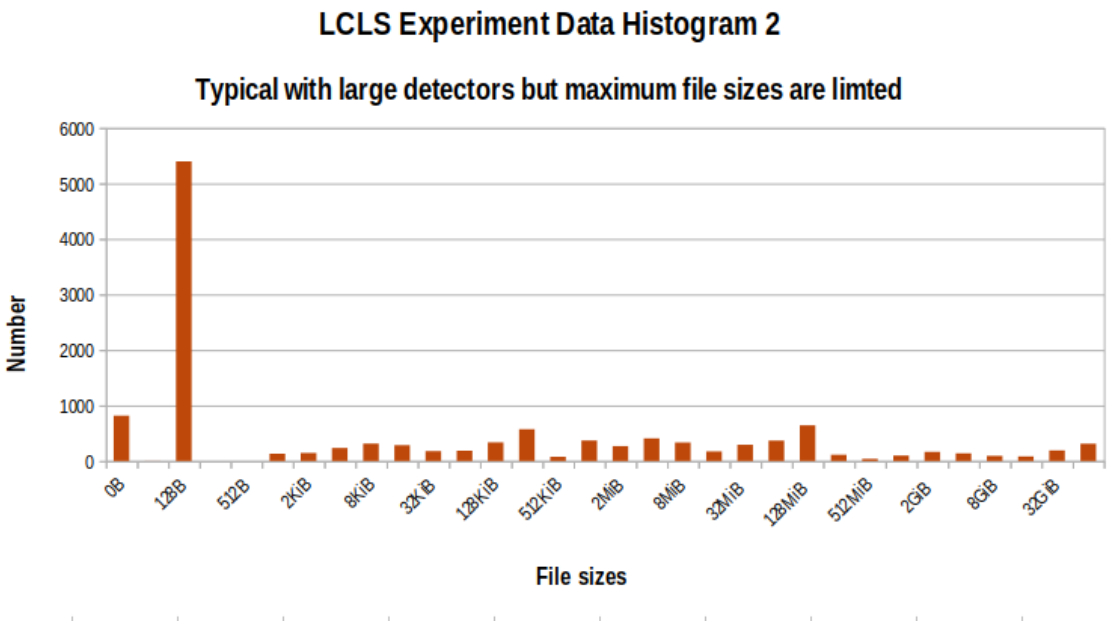

FIG. 18 The same observation for **FIG. 17** applies here too

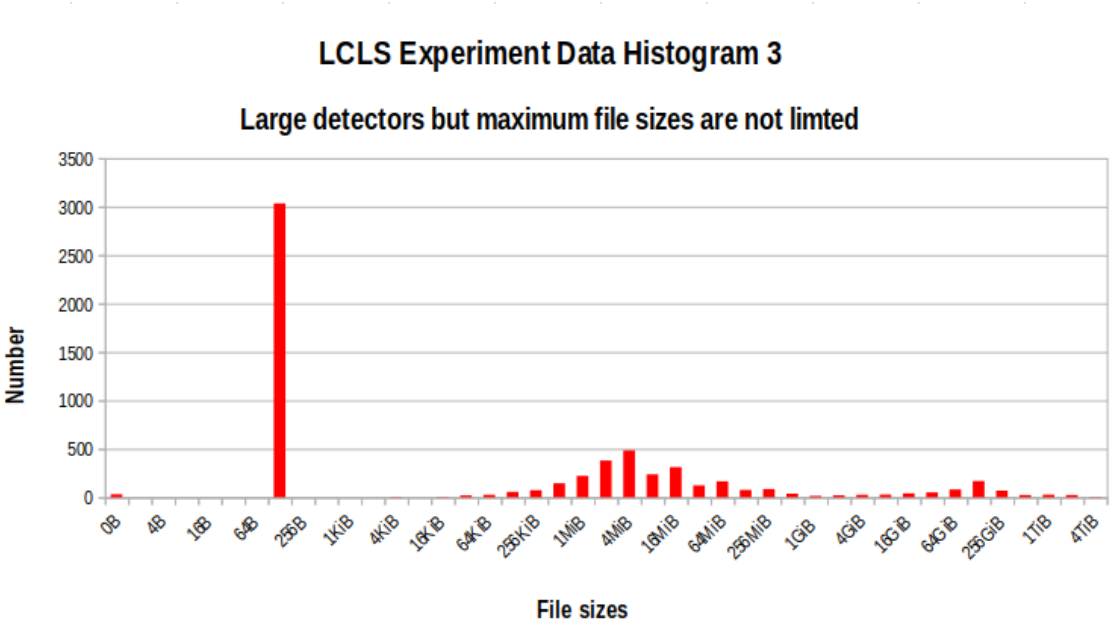

FIG. 18 Note that a few of the files generated for this LCLS experiment have sizes around 2TiB and 4TiB

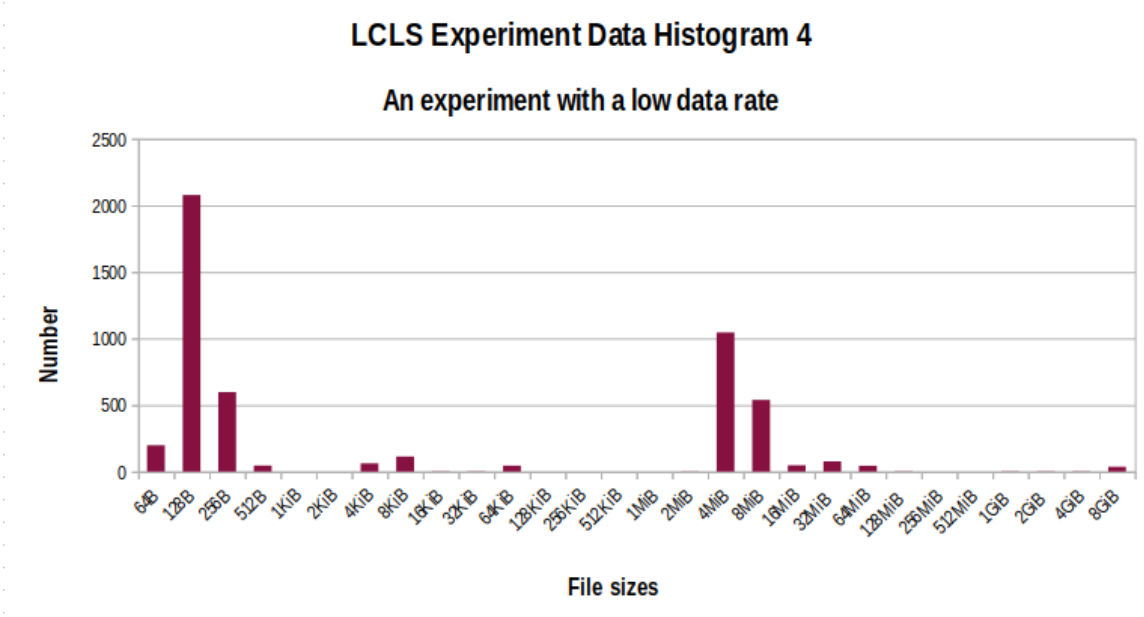

FIG. 18 The file size distribution for this LCLS experiment tend to cluster around two sizes

Conclusions

1. Rsync and rsync-based tools should be used only for casual data movement needs. In other words, they are tools for Category 4 listed in reference 1.
2. Using a rsync wrapper such as parsyncfp is not as simple as one may expect. The installation, configuration may exceed many users' computer competency. For example, parsyncfp and its dependencies are not available in ready-to-use packages for major Linux distros. The error handling may catch many by surprises.
3. Based on our test outcomes, we suggest rsync to be used for moving files in at most metro distance (RTT ≤ 10ms) and for datasets that are less than 1TB in overall size, with files sized in the upper side of the LOSF range (≥ 128KiB) or the medium range (≤ 1GiB).

Beyond what is outlined above, we recommend using a PDDM.

Acknowledgments

We gratefully acknowledge the review and thoughtful comments from all the reviewers: Alan Charles Davis, National University of Singapore[25]; Sandra Wambold; Martin Buckholz; Sven Breuner; Wilko Kroeger, SLAC National Accelerator Laboratory; David Skinner, National Energy Research Scientific Computing Center (NERSC)[26]; Andrew Howard, National Computing Infrastructure[27], Australia; Mark Gray, Pawsey Supercomputing Center[28], Australia.

References

1. Chin Fang, Les Cottrell, "Data Movement Categories". The U.S. DOE Technical Report OSTI ID 1756618, https://www.osti.gov/biblio/1756618. Accessed March 12, 2021
2. Office of Science User Facilities, https://www.energy.gov/science/science-innovation/office-science-user-facilities. Accessed March 21, 2021
3. Andrew Tridgell. https://en.wikipedia.org/wiki/Andrew_Tridgell. Accessed March 12, 2021
4. Andrew Tridgell, "Efficient Algorithms for Sorting and Synchronization", Ph.D. Dissertation, The Australian National University, February 1999. https://www.samba.org/~tridge/phd_thesis.pdf. Accessed March 12, 2021
5. "rsync - a fast, versatile, remote (and local) file-copying tool". https://download.samba.org/pub/rsync/rsync.1. Accessed March 12, 2021
6. Harry Mangalam et.al. "parsyncfp, a parallel rsync wrapper in Perl". https://github.com/hjmangalam/parsyncfp. Accessed March 12, 2021

[25] https://www.nus.edu.sg
[26] https://www.nersc.gov
[27] https://nci.org.au
[28] https://pawsey.org.au

7. Navin Shenoy, "Innovating for the 'Data-Centric' Era", Intel Innovation Summit presentation, August 8, 2018, https://newsroom.intel.com/editorials/data-centric-innovation-summit/. Accessed March 12, 2021
8. "Identifying ZFS Snapshot Differences (zfs diff)", https://docs.oracle.com/cd/E36784_01/html/E36835/gkkqz.html. Accessed March 12, 2021
9. Red Hat Enterprise Linux 7, Storage Administration Guide, "14.3.2. Comparing Changes with the diff Command", https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/7/html/storage_administration_guide/ch-snapper-status-command#snapper-diff. Accessed March 12, 2021
10. "Zettar Inc.'s testbed", https://www.youtube.com/watch?v=5qTpGg57p. Accessed March 12, 2021
11. Ezra Kissel, Chin Fang, "Zettar zx Evaluation for ESnet DTNs", https://www.es.net/assets/Uploads/zettar-zx-dtn-report.pdf. Accessed March 12, 2021
12. Chin Fang, github test_rsync repo, https://github.com/fangchin/test_rsync. Accessed March 14, 2021
13. Chin Fang, "High-Performance Data Movement Services - DTNaaS", Rice 2021 Oil & Gas High-Performance Conference Technical Program Lightning talk, March 5, 2021, https://youtube.com/watch?v=f5C2b7aYlnk. Accessed March 12, 2021
14. Chin Fang, "elbencho storage sweep tools", https://github.com/breuner/elbencho/tree/master/contrib/storage_sweep. Accessed March 12, 2021
15. NEWS for rsync, https://download.samba.org/pub/rsync/NEWS#3.2.3, "rsync-ssl should be considered as mainstream now that Samba requires SSL for its rsync daemon". Accessed March 12, 2021.
16. "rsyncd.conf(5) — Linux manual page", https://download.samba.org/pub/rsync/rsyncd.conf.5.html. Accessed March 12, 2021
17. The U.S. Department of Energy, Office of Science. https://www.energy.gov/science/office-science. Accessed March 21, 2021
18. "Globus Subscriptions", https://www.globus.org/subscriptions. Accessed March 12, 2021
19. "Zettar Inc. Resources", https://www.zettar.com/resources/. Accessed March 12, 2021
20. "Welcome to the official home of the Lustre® filesystem", https://www.lustre.org. Accessed March 12, 2021
21. slurm workload manager, "Overview", https://slurm.schedmd.com/overview.html. Accessed March 12, 2021
22. JOINT GENOME INSTITUTE, A DOE OFFICE OF SCIENCE USER FACILITY, https://jgi.doe.gov. Accessed March 12, 2021
23. LCLS, Linac coherent Light Source. https://lcls.slac.stanford.edu. Accessed March 12, 2021
24. Sven Breuner, Chin Fang, "elbencho - A new Storage Benchmark for AI et al", PPoPP'21 Workshop: Benchmarking in the Data Center", https://parallel.computer/presentations/PPoPP2021/elbenchoANewStorageBenchmarkForAIetal.pdf. Accessed March 12, 2021
25. NUS, National University of Singapore. https://www.nus.edu.sg. Accessed March 20,2021

26. National Energy Research Scientific Computing Center. https://nersc.gov. Accessed March 20, 2021
27. NCI Australia. https://nci.org.au. Accessed March 20, 2021
28. Pawsey Supercomputing Center. https://pawsey.org.au/. Accessed March 20, 2021

Appendix

A.1 The basis for the postulation

Explained in this appendix is the basis of the postulation posed in the section **A few important factors and a postulation**. Note that regardless of other differences, the clock speed of ESnet testbed nodes' CPUs is **23%** higher than the ones used for Zettar testbed nodes. Note that rsync is single threaded, so the difference in the number of hyperthreads (aka logical cores), unless an aggregation based approach is used (e.g. such as by parsyncfp) is not that important.

Also, compare Zettar testbed and ESnet LAN parsyncfp results carefully and keep the attained levels for different file sizes shown in **FIG. 1** and **FIG. 2** in mind.

Zettar testbed nodes have the following CPU:

```
[root@zh0 ~]# lscpu
Architecture:          x86_64
CPU op-mode(s):        32-bit, 64-bit
Byte Order:            Little Endian
CPU(s):                48
On-line CPU(s) list:   0-47
Thread(s) per core:    2
Core(s) per socket:    12
Socket(s):             2
NUMA node(s):          2
Vendor ID:             GenuineIntel
CPU family:            6
Model:                 85
Model name:            Intel(R) Xeon(R) Gold 6126 CPU @ 2.60GHz
Stepping:              4
CPU MHz:               999.914
CPU max MHz:           3700.0000
CPU min MHz:           1000.0000
BogoMIPS:              5200.00
Virtualization:        VT-x
L1d cache:             32K
L1i cache:             32K
L2 cache:              1024K
L3 cache:              19712K
NUMA node0 CPU(s):     0-11,24-35
NUMA node1 CPU(s):     12-23,36-47
```

ESnet testbed nodes have the following CPU:

```
[root@nersc-tbn-7 ~]# lscpu
Architecture:          x86_64
CPU op-mode(s):        32-bit, 64-bit
Byte Order:            Little Endian
CPU(s):                24
On-line CPU(s) list:   0-23
Thread(s) per core:    1
Core(s) per socket:    12
Socket(s):             2
NUMA node(s):          2
Vendor ID:             GenuineIntel
CPU family:            6
Model:                 85
Model name:            Intel(R) Xeon(R) Gold 6146 CPU @ 3.20GHz
Stepping:              4
CPU MHz:               3899.976
CPU max MHz:           4200.0000
CPU min MHz:           1200.0000
BogoMIPS:              6400.00
Virtualization:        VT-x
L1d cache:             32K
L1i cache:             32K
L2 cache:              1024K
L3 cache:              25344K
NUMA node0 CPU(s):     0-11
NUMA node1 CPU(s):     12-23
```