elbencho

A new Storage Benchmark for AI et al

PPoPP'21 Workshop:
Benchmarking in the Data Center

Sven Breuner
sven.breuner@gmail.com

Chin Fang
fangchin@zettar.com

Background

Who?

Sven Breuner.

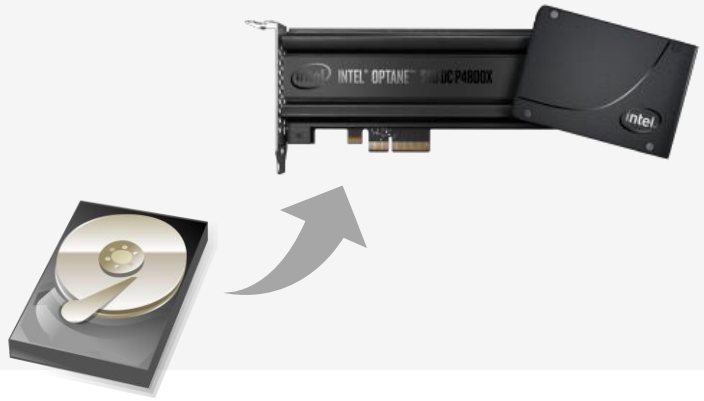
Creator of the **BeeGFS** parallel file system.

Nowadays focus on all-flash storage.

Why?

Understanding storage system characteristics is key to data analytics **efficiency & scaling**.

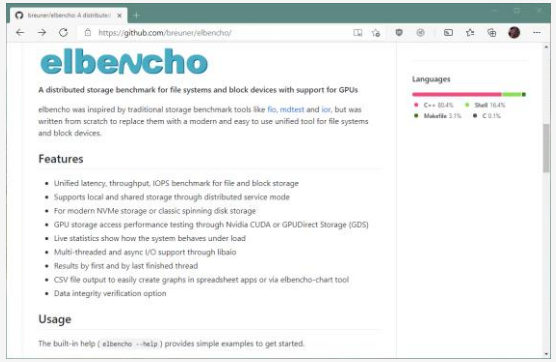
File system characteristics cannot be derived from hardware specs.



Where?

GitHub

https://github.com/breuner/elbencho



What are typical storage metrics of interest?

Depending on workload and data format…

1. Especially in Deep Learning image recognition:
   Lots of small file reads per second
   -or-
   Small random reads IOPS in large files

2. For databases:
   Access latency

3. For HPC:
   Streaming
   -or-
   Shared file writes

…all with high concurrency and typically on shared storage.

Flash storage embraces high concurrency, but too much concurrency can have negative side effects (e.g. on the CPU).

What can elbencho show you?

All the metrics of interest ☺

from a single client or coordinated across multiple clients

```
$ elbencho /mnt/smb --hosts devel1,devel2
  -r -t3 -n10 -N30 -s128k -b128k --lat --direct

OPERATION RESULT TYPE         FIRST DONE  END RESULT
========= ================    ==========  ==========
READ      Elapsed seconds   :         12          14
          Files/s           :        130         127
          IOPS              :        130         127
          Throughput MiB/s  :         16          15
          Total files       :       1631        1800
          Total MiB         :        203         225
          Files latency ms  : [ min=2 avg=44 max=565 ]
          IO latency ms     : [ min=1 avg=35 max=548 ]
```

Hint: Distributed runs are easy (without MPI)

```
devel1:~$  elbencho --service
devel2:~$  elbencho --service
master1:~$  elbencho --hosts devel1,devel2 ...
# Or alternatively:
master1:~$  elbencho --hostsfile myhosts.txt ...
```

Live Statistics

```
Phase: READ  CPU: 1%  Active: 2  Elapsed: 10s
 Rank    %  DoneMiB   MiB/s   IOPS   Files  Files/s  Act CPU Service
Total   68       153      14    117    1224      117    6   1
    0   68        77       7     61     616       61    3   1 devel1
    1   67        76       7     56     608       56    3   1 devel2
```

How to use elbencho?

Lots of small files
Benchmark path is a directory

```
$ elbencho /mnt/smb/mydir
  -r -t3 -n10 -N30 -s128k -b128k --direct

OPERATION RESULT TYPE         FIRST DONE  END RESULT
========= ================    ==========  ==========
READ      Elapsed seconds   :         12          14
          Files/s           :        130         127
          Throughput MiB/s  :         16          15
          Total files       :       1631        1800
          Total MiB         :        203         225
```

- **-r / -w** Read or write files
- **-t** Number of threads
- **-n** Number of directories per thread
- **-N** Number of files per directory
- **-s** File size

Random read IOPS
Benchmark path is a directory

```
$ elbencho -w -s 50g /mnt/tmpfs/myfile

$ elbencho -r -t 4 -b 4k --lat --direct --rand --timelimit 10
  /mnt/tmpfs/myfile

OPERATION RESULT TYPE         FIRST DONE  END RESULT
========= ================    ==========  ==========
READ      Elapsed seconds   :         10          10
          IOPS              :     130214      130214
          Throughput MiB/s  :        509         509
          IO latency ms     : [ min=1 avg=35 max=548 ]
```

Time limit (in seconds) can be used to avoid long wait times that won't change the IOPS result

Bonus Feature #1: elbencho for GPUs

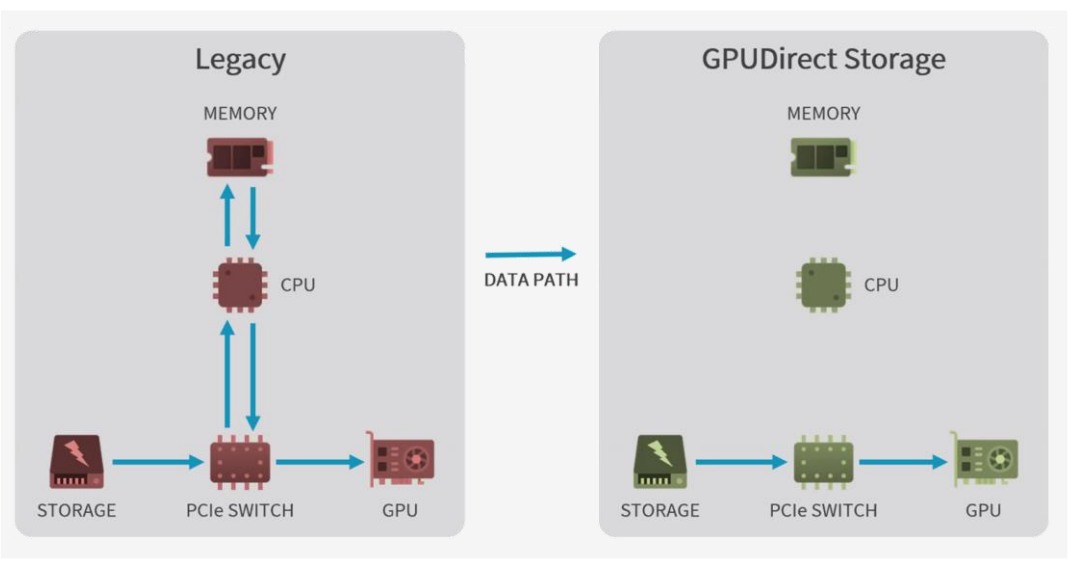
GPU data transfer via CUDA

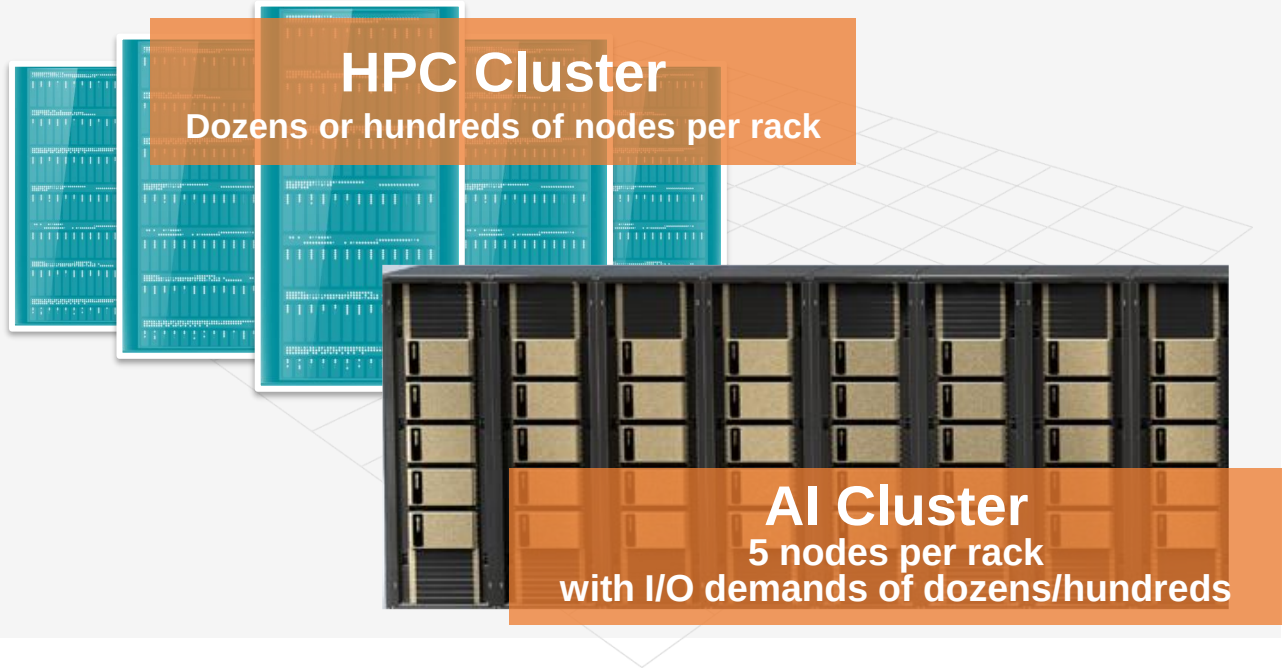
```
# 1MB random reads via host memory into GPU memory
dgx-a100$ elbencho -t 128 -r -s10g -b 1m --direct -rand
    /data/file{1..128} --gpuids "0,1,2,3,4,5,6,7" --cuhostbufreg
Result: 45.7GB/s

# Read 512000 small 128KiB files via host memory into GPU memory
dgx-a100$ elbencho -t 128 -r --direct -n 40 -N 100 -s 128k
    /data --gpuids "0,1,2,3,4,5,6,7" --cuhostbufreg
Result: 139444 files per sec
```

GPUDirect Storage (GDS)

```
# Using cuFile API for GDS
dgx-a100$ elbencho --cufile --gpuids "0,1,2,3,4,5,6,7" ...
```





Bonus Feature #2: elbencho charts

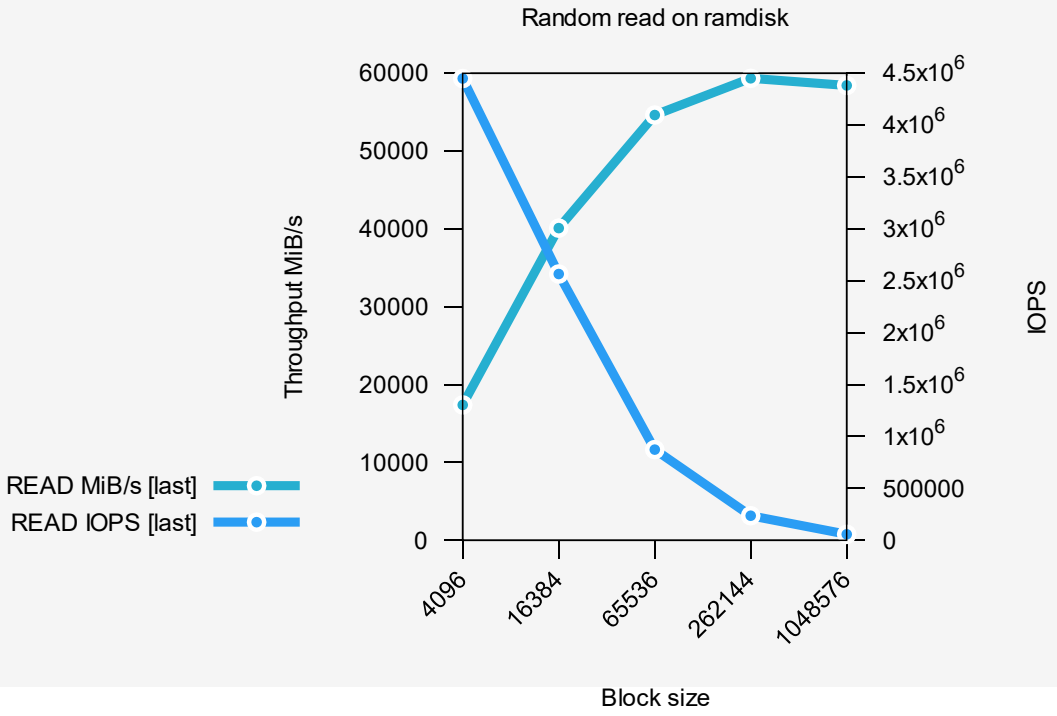
CSV file output

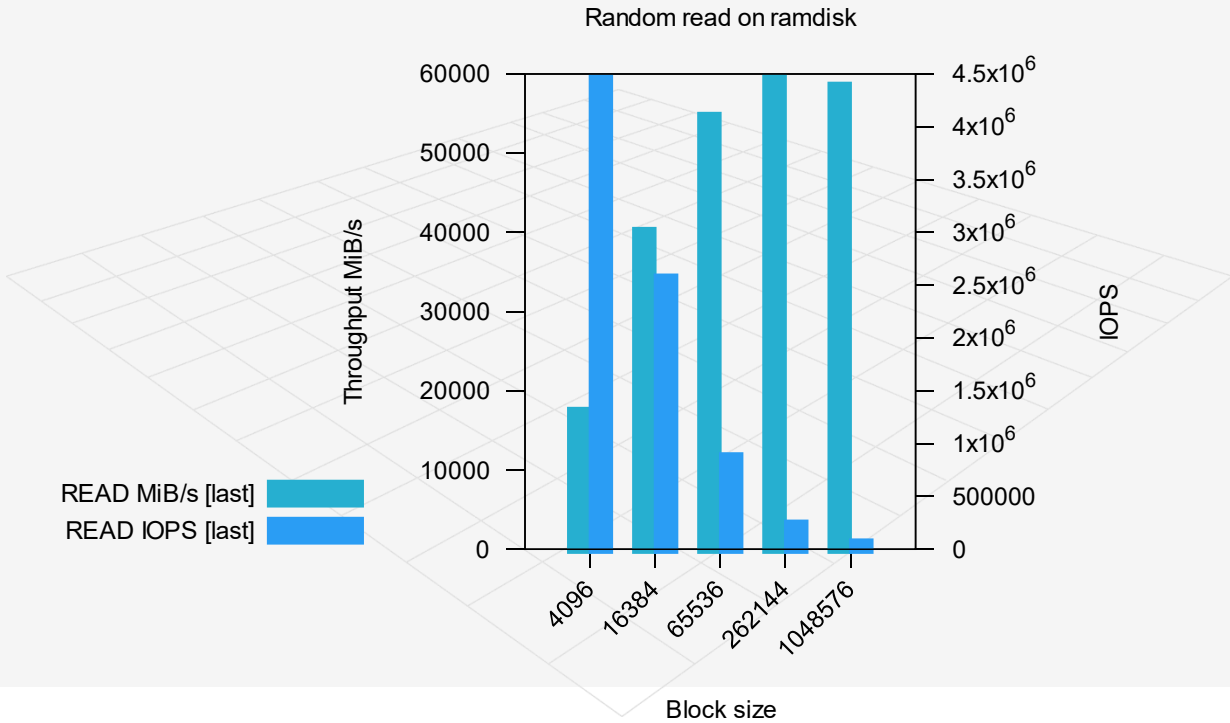
```
$ for block in 4k 16k 64k 256k 1m;
  do
    elbencho --csvfile results.csv -t 4 -r -b $block
       /mnt/tmpfs/file --timelimit 10 --rand
  done
```

elbencho-chart tool

```
# Generate line chart from CSV file
$ elbencho-chart -x "block size"
  -y "MiB/s [last]" -Y "IOPS [last]" results.csv

# Bar charts are also possible
$ elbencho-chart --bars ...
```





Storage Sweep

Who?

Chin Fang.

Founder and CEO of Zettar Inc.

Zx: a Universal Data Mover for moving data at scale and speed

Why?

Enabling the following:

- Understand a storage service quickly and simply
- Pick the most performant entry from all candidates easily and accurately
- Evaluate the impact of a tuning approach
- Many more

Where?

GitHub

https://github.com/breuner/elbencho/*contrib*/storage_sweep/


README.md

December 25, 2020

Storage Sweep

Chin Fang < fangchin[at]zettar.com >, Palo Alto, California, U.S.A

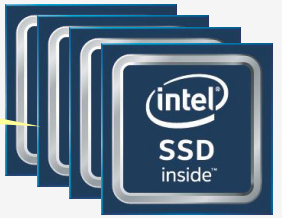
Programming is Gardening, not Engineering

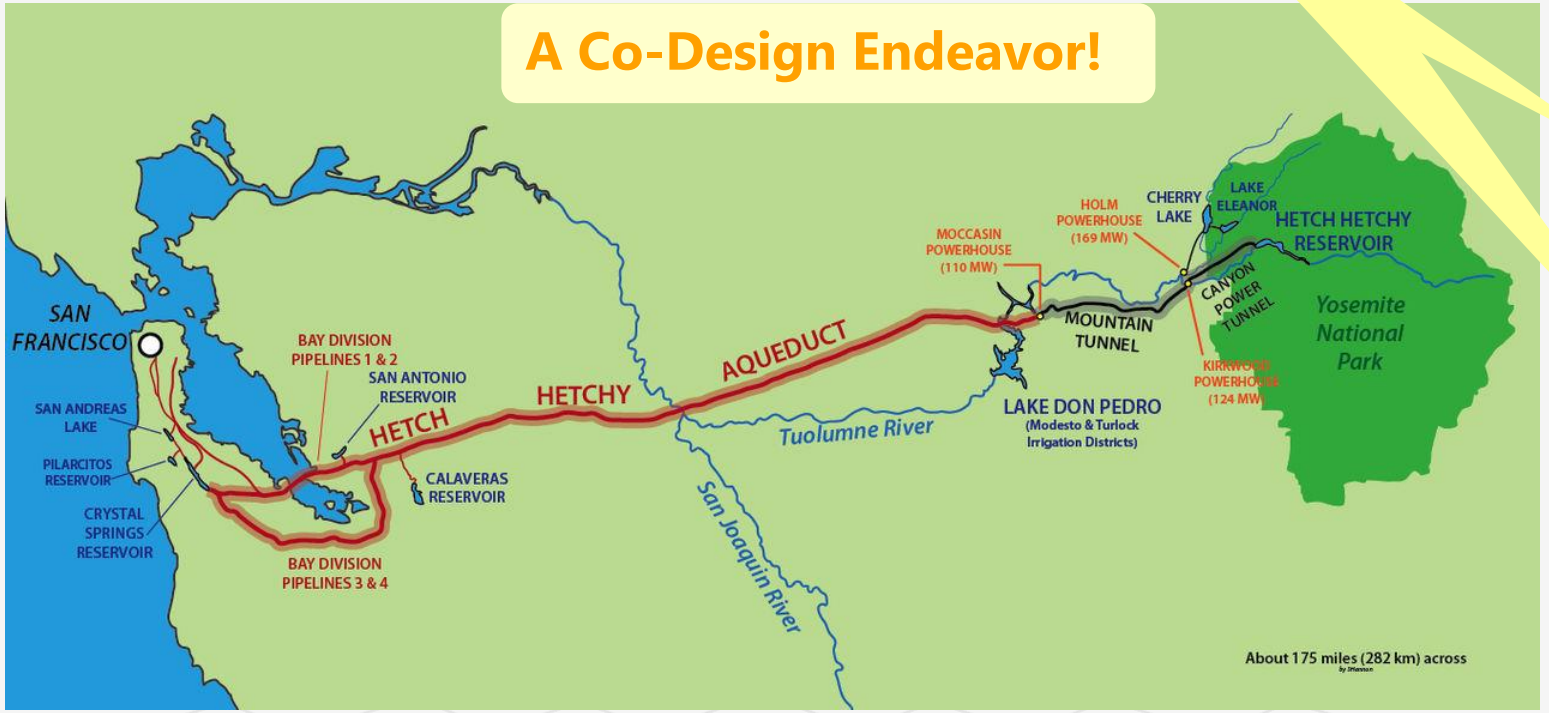
Table of contents

- Introduction
- Requirements
- Layout and content
- Goals and possible uses
  - Primary goal of each wrapper and the main use
    - mtelbencho.sh
    - graph_sweep.sh
  - Other uses
- Motivation
- A brief overview of storage benchmarking
  - A typical goal
  - How it should be done
  - A common misconception
  - Carry out your own storage benchmarking
- Future evolution
- Acknowledgments
- Epilogue

Moving data at scale and speed & storage? A water transport analogy

Get these ready first!





Picture credit: Wikipedia

The 1st Step: Storage I/O Benchmarking - I

Or

elbencho Storage Sweep

1. Choices of tools, fio?
2. Recommendations?
3. Methodology?
4. What to do?
5. Why?

master · 2 branches · 11 tags · Go to file · Add file · Code

fangchin contrib: polished storage_sweep (former mtelbencho) by Zettar's Chin ... f81e5af 9 hours ago 76 commits

bin	all: initial commit	6 months ago
build_helpers	make: fix typos in Makefile related to auto-detection of CUDA path	2 months ago
contrib/storage_sweep	contrib: polished storage_sweep (former mtelbencho) by Zettar's Chin ...	9 hours ago
dist/etc/bash_completion.d	rwmix: add new option for mixed read+write	9 days ago
external	communication: update embedded http server to latest version	2 months ago
packaging	gds: update path detection for GPUDirect Storage v0.9 beta	2 months ago
scripts	worker: change elapsed time res from milli to microsecs	4 months ago
source	rwmix: add new option for mixed read+write	9 days ago
.gitignore	args: add bash completion support	2 months ago
CHANGELOG.md	rwmix: add new option for mixed read+write	9 days ago
LICENSE	Initial commit	6 months ago
Makefile	rwmix: add new option for mixed read+write	9 days ago
README.md	comments: fixed minor typos in comments	last month

README.md

elbencho

About

A distributed storage benchmark for file systems and block devices with support for GPUs

benchmark · storage · nvme · deep-learning · live-stats · distributed · parallel · file-systems · block-storage · ior · mdtest · fio · gpu

Readme

GPL-3.0 License

Releases

11 tags

Packages

No packages published

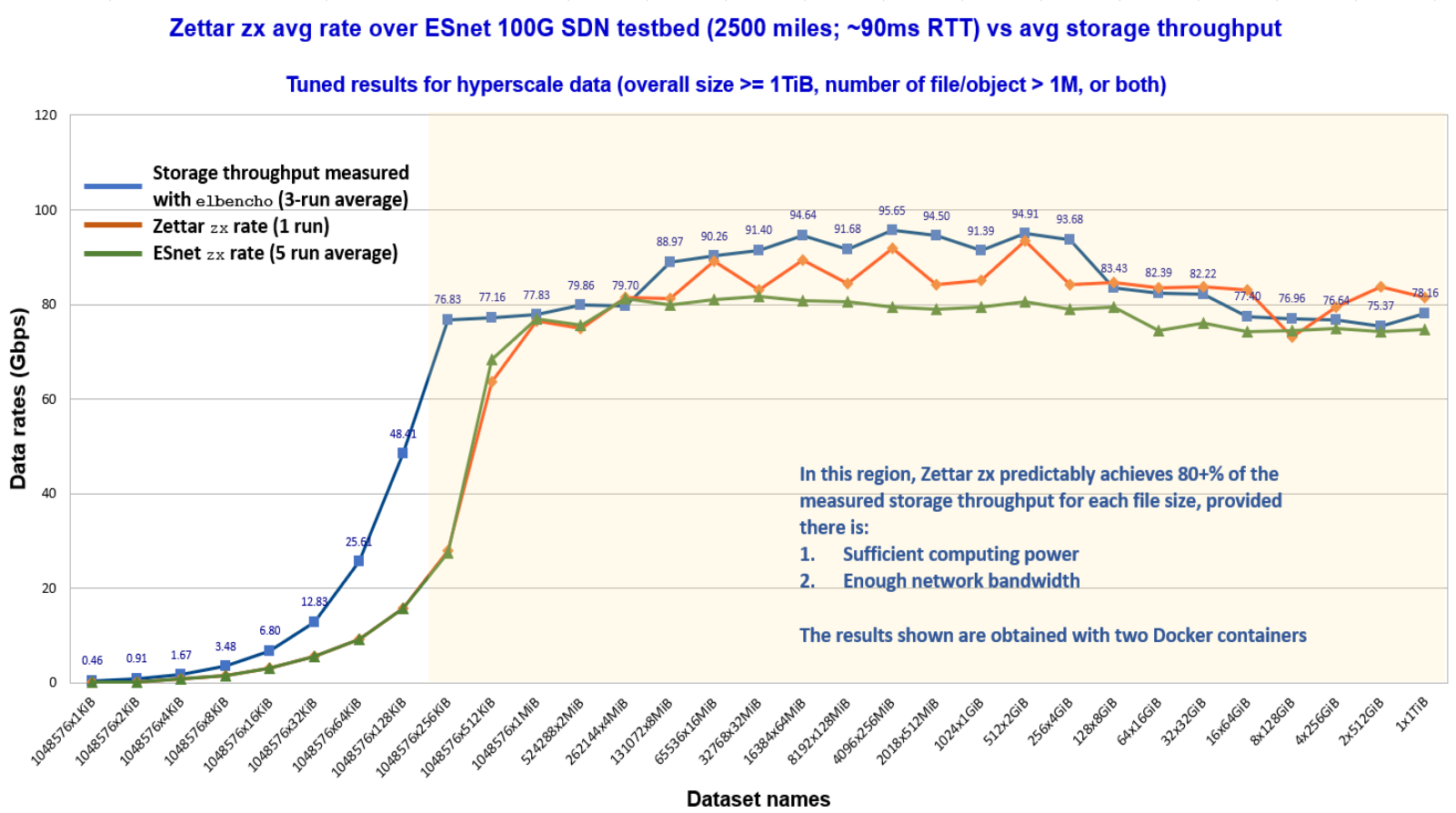
Contributors 2

breuner Sven Breuner

fangchin Chin Fang

The 1st Step: Storage I/O Benchmarking - II

1. 1st: production dataset
2. 2nd: storage sweep
3. An example

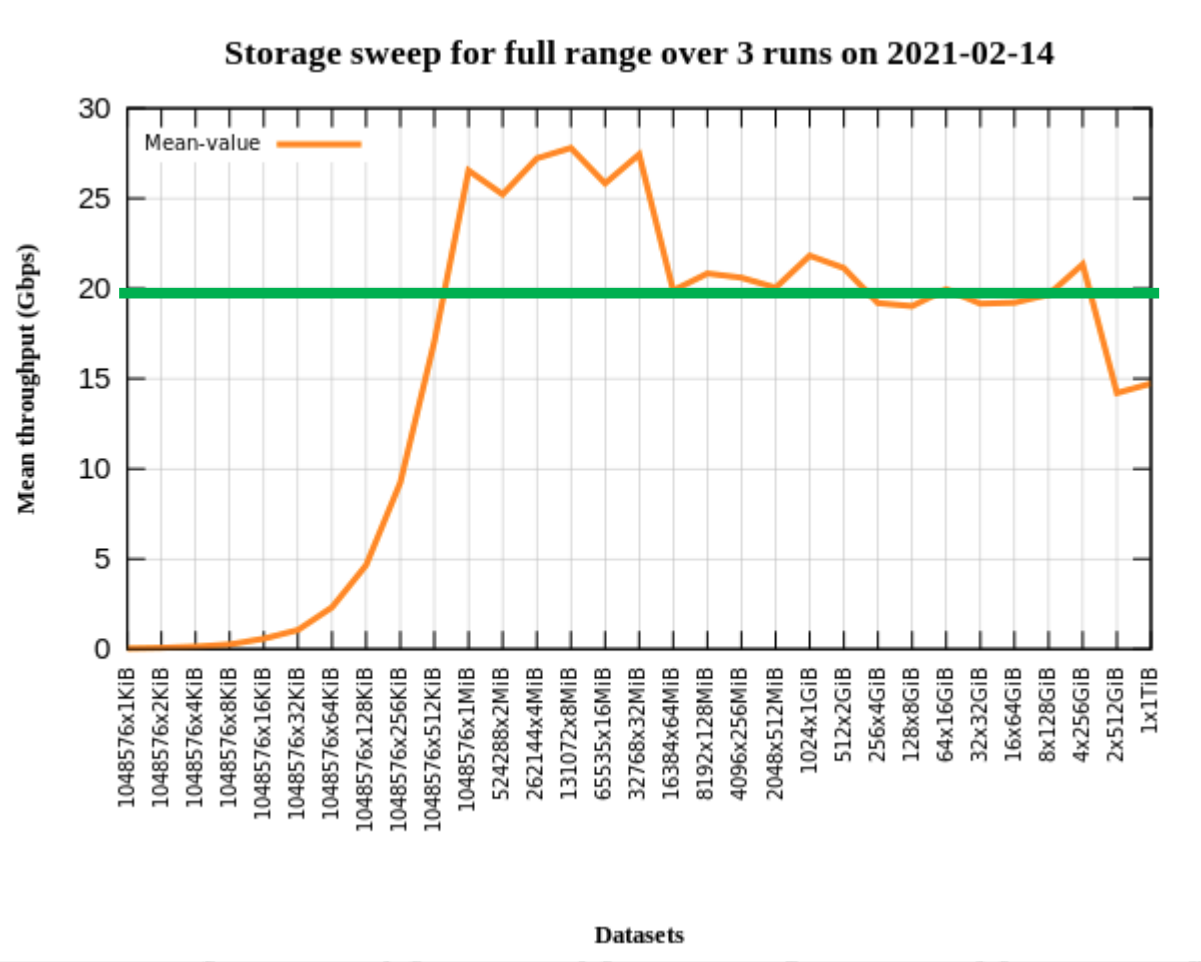


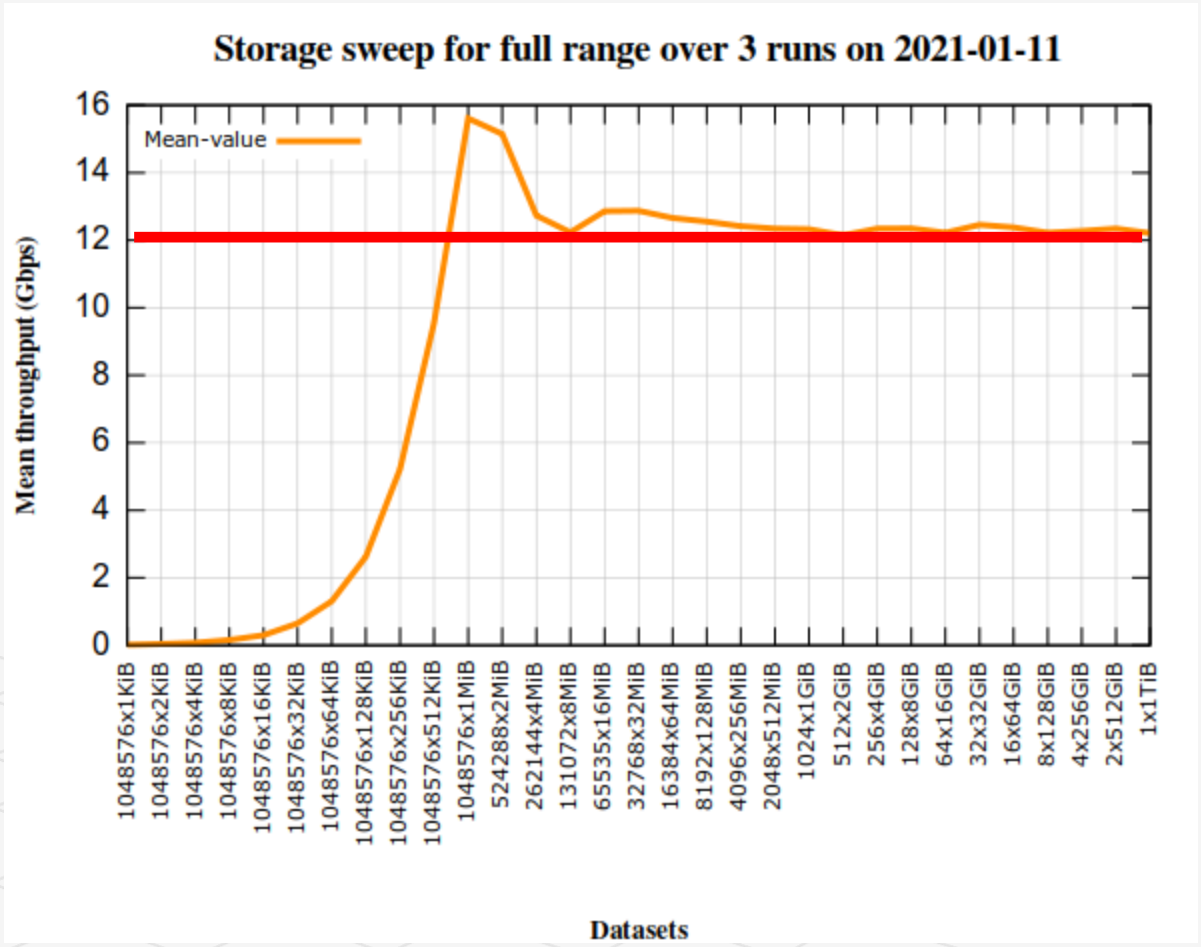
A combined line chart with the measured storage throughput for each file size (**blue line**), together with both the Zettar zx transfer data rates attained with a single run carried out by Zettar (**orange line**), and the average of five runs carried out by ESnet (**green line**). The X-axis labels are the test dataset names. For example, **2x512GiB** means the dataset has **2** files, each is **512GiB** in size. Thus, the dataset has an overall size **1TiB** = **2 x 512GiB**

Some references

1. **Chin Fang, Les Cottrell, Data Movement Categories**.
   https://www.osti.gov/biblio/1756618
2. **Ezra Kissel, Chin Fang, Zettar zx Evaluation for ESnet DTNs.**
   https://bit.ly/3pG4H24
3. **Ezra Kissel, 100G DTN Experiment: Testing Technologies for Next-Generation File Transfer.**
   https://bit.ly/3qfJi0g

What does storage sweep tools do for you?



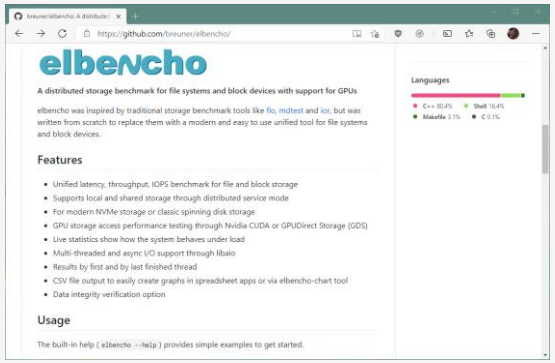


Now what?

Download, understand your system and then
run your workload most efficiently

GitHub

https://github.com/breuner/elbencho



Share feedback or contribute

Sven Breuner
sven.breuner@gmail.com

Chin Fang
fangchin@zettar.com