Terabyte Transfers over High-BDP Links

Tactics and Architecture

small fixes now, right shapes later

XRootD / FTS Workshop; Lyon – 14–18 Sept 2026

J. Walder, STFC-RAL / ukSRC

The problem

- Given a set of arbitrarily sized files (data product);
  e.g. 5TB files, over 300ms RTT links

- Constraint:
  that the final state of the files on disk ***is (or looks like)*** the original file

- Is WLCG TPC transfer strategy sufficient to meet this need

  - Where are the places that changes are needed, with least effort / divergence?

- If not TPC; what can still exist below the Rucio layer?

The transfer that worked

RAL → SWE Chalmers · one file · HTTP third-party copy

File ID	File State	File Size	Throughput	Remaining	Start Time	Finish Time	Staging Start	Staging End	Archiving Start	Archiving End	
102570731	FINISHED	9313.23 GiB	236.76 MiB/s	–	2026-08-26T18:21:52Z	2026-08-27T05:34:05Z	–	–	–	–	Log

https://xrootd01.uksrc.rl.ac.uk:1094/uksrc-ral-t1/srcnet_ops.comm/eb-001.random-10TB-1/random-10TB-aa38c527

davs://xrootd-01.swesrc.chalmers.se:1094/data/rse/jwtests/tc/campaign_april_2026/testfile_f71146e47c0547239e3943b35af1e5b1

10 TB

one file

1.85 Gb/s

236.76 MiB/s — 1.8 % of a 100G path

~11hrs

wall clock, first attempt

Single test worked (small BDP);

so did the 20x 10x5TB = 1PB file run (no failures; short link)

Bandwidth delay product

Double cost: Window size and duration

Problem 1 — the window size

- Telescope to SRCNet Nodes could be up to O(300)ms RTT.
- 100 Gb/s × 300 ms ⇒ 3.75 GB in flight
- Per-stream ceiling ~ 30 Gb/s
  (1 GiB window ÷ 0.3 s, wscale 14)
- Loss-based CC (Reno, CUBIC) collapses with loss at this BDP
- **BBR at the sender can mitigate some of this:**

Mostly solved – on well-tuned systems;
problem 2 is the subject of the talk.

Problem 2 — the duration

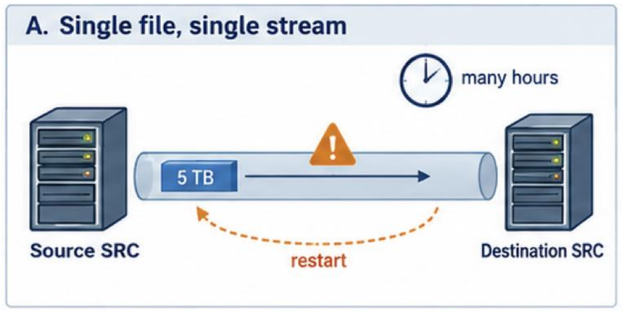
$P(\text{no restart}) = (1 - p)^T$ p = disruptions per transfer-hour, T = duration

p / hour	3 h 100 GB @ 15 MB/s	11 h 10 TB @ 1.85 Gb/s	48 h	110 h 100 TB	efficiency @ 11 h
0.2 %	99 %	98 %	91 %	80 %	99 %
0.5 %	99 %	95 %	79 %	58 %	97 %
1 %	97 %	90 %	62 %	33 %	95 %
2 %	94 %	80 %	38 %	11 %	89 %

● shaded: WLCG tail today · SRCNet file today.

Throughput gives reliability. Duration impacts all.
p covers the random failures only —

Where do communities sit?



LHC typical file
few GB @ 15 MB/s ≈ 5 min

LHC tail: 100 GB
@ 10–20 MB/s = 1.4–2.8 h

SRCNet now: 5 TB @ 20–250 MB/s
0.16–2 Gb/s → 5.6–69 h

> 24 h: p stops mattering
resume is mandatory

measured
9.3 TiB @ 1.85 Gb/s = 11 h 12 m

size trend

5 min | 30 min | 1 h | 3 h | 11 h | 24 h | 110 h

transfer duration (log)

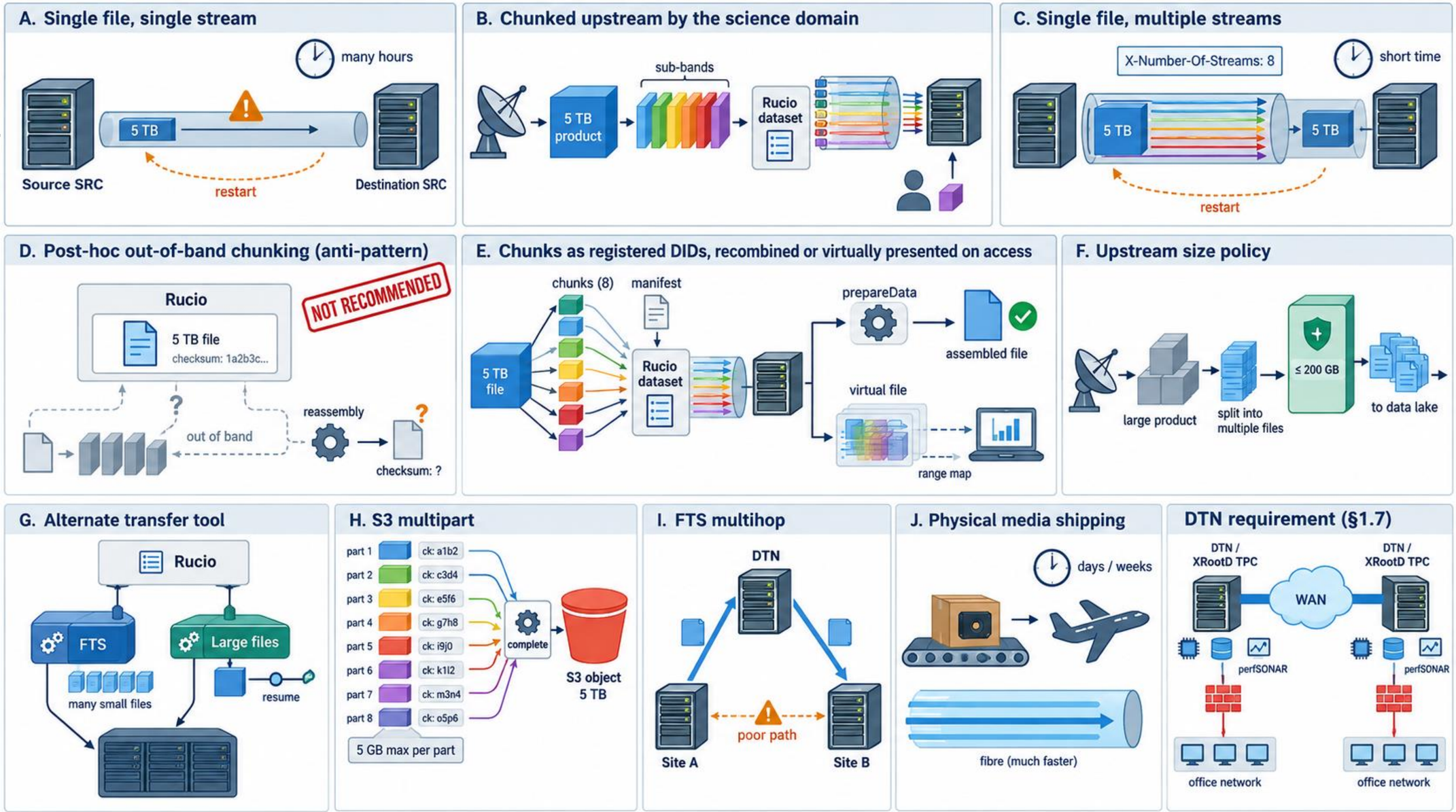
tactics: BBR, streams
5 TB @ 25 Gb/s ≈ 27 min

SRCNet next: 100 TB
@ 2 Gb/s ≈ 110 h — and that is the fast end

WLCG: not an efficiency problem today — a token and last-file-in-the-dataset problem today, a size problem in a few years. **SKA:** gets there by file size. **Same fix.**

Transfer strategies

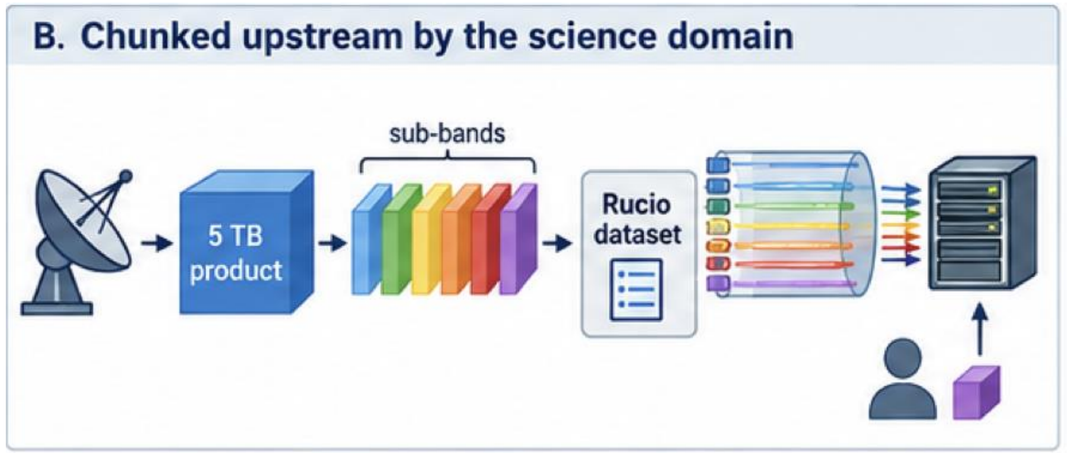
- Tl



Architecture: the right design?

Chunk upstream in science-meaningful units — when you can

- **Sub-bands, time ranges, tiles — units a user would ask for anyway**
  - **The retry unit becomes smaller than the failure interval**
  - **Scheduling and retry are handed to FTS + Rucio, which already do this well**
- **MSv4 / Zarr / xarray are built this way.**
  MSv2 is a directory tree — the single file is a packaging choice (tar)
- **Not available to legacy products, and not in the short term for current→ we need a tactic meanwhile**



5 TB

hard ceiling on a single object in S3-style stores — the strongest argument for chunking that is not about the network

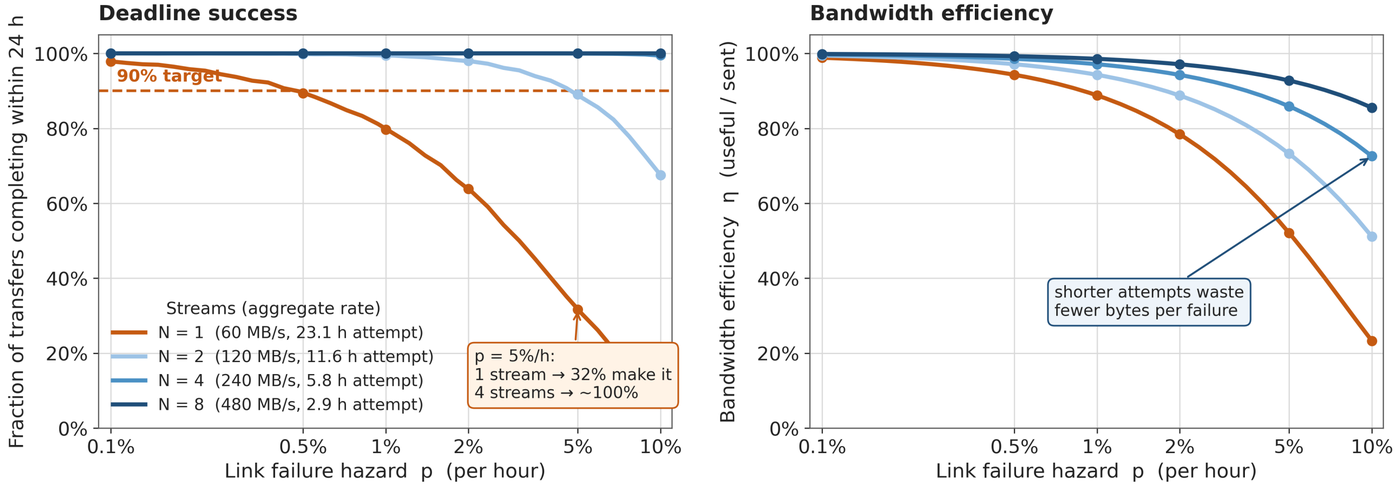
(Resumable) Multi-stream TPC

What to do now?

- Multi-stream, mitigates the duration-related issues.
- On reasonable links; files complete within time, and minimal loss of efficiency.
- With resumability, even then on poor / slow links, transfers go to completion.

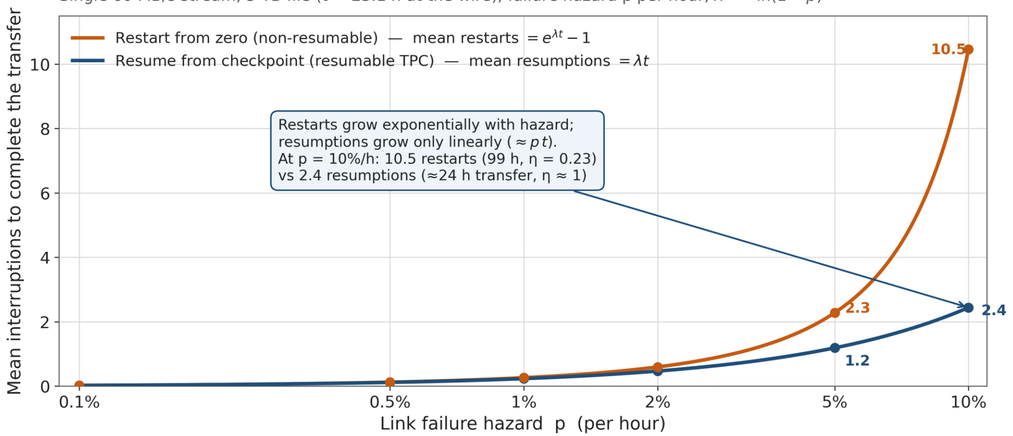
Parallel streams shrink the exposure window: 5 TB over N×60 MB/s TCP streams

Whole-path hazard p per hour; any failure aborts the transfer, which restarts from zero. N streams raise the aggregate rate, cutting the hours at risk (survival = $(1-p)^t$, t ∝ 1/N).



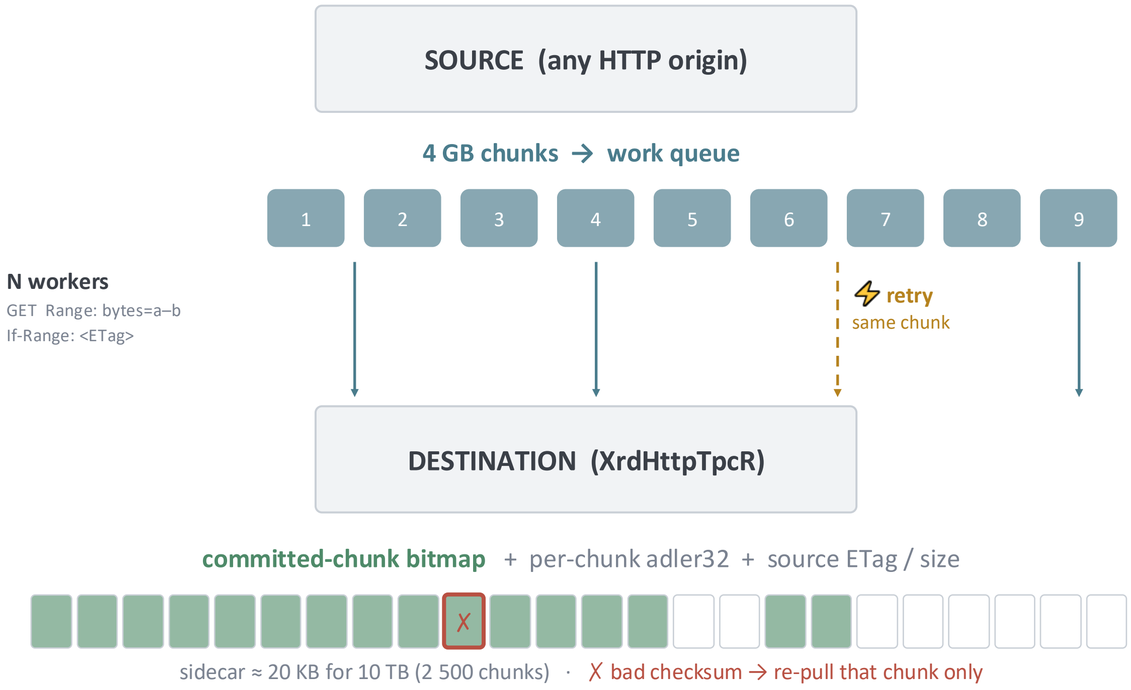
Resumable transfers turn an exponential problem into a linear one

Single 60 MB/s stream, 5 TB file (t = 23.1 h at the wire); failure hazard p per hour; $\lambda = -\ln(1-p)$



TPCR: resumable TPC-pull

Large-file tactics: multi-stream, drop-in for XrdHttpTpc



Resume trigger = the FTS retry / internal retry.

Destination finds a matching partial (same ETag/size) → skips done chunks. FTS brings the fresh token.

- **Chunks, say 4 GB,** pulled from a queue: streams self-balance, loss per failure ≤ one chunk per worker
- **adler32 per chunk on the write path,** adler32_combine in chunk order → file checksum with no read-back.
- **State = bitmap + per-chunk sums + ETag/size.** Sidecar (or xattr where large values are allowed). Never deleted on failure ☹
- **XRootD owns the bytes, FTS owns the retry.** Token lifetime solved for free; FTS changes to one thing: don't purge the partial
- If not this; XRootD needs to do the retry (token lifetime?)
- **Needs multi-stream hardened**
  (X-Number-Of-Streams is implemented, lightly used) —
  SRCNet will need this path first

- Untested idea on github XrdHttpTpc**R**

Assembly, not invention

Every ingredient exists; the work is wiring, not design

EXISTS

Range + If-Range

RFC 9110. Resume a GET from any offset; if the source changed you get 200 and start over. Every HTTP origin shoud speak it.

EXISTS

adler32_combine

zlib, since forever. Per-chunk sums on the write path combine into the file checksum Rucio and FTS already accept.

EXISTS

FTS retry?

The retry loop is the resume trigger and the token refresher. One behaviour change: keep the partial.

Limited, if keep retry within xrootd itself.

Who

SRCNet needs it

5 TB files, 300 ms paths, multi-stream switched on. We will be the first to exercise it hard — and report back.

Resumable TPC-pull = these three, wired together.

Push direction: IETF draft-ietf-httpbis-resumable-upload (tus-derived) is the twin — not needed for pull.
Possible solution for Rucio upload say

Summary

Advancing TPC into the next-generation of Data Transfers

- High-latency BDP paths and large file sizes present a challenge to new communities
- Does (WebDAV) TPC have a place in this environment; if so, how to embrace it
- Optimization of file size and format critical (main design win); not always possible
- Find (and fix) fragile behaviour in current X-Number-Of-Streams multi-stream code path
- Resumable transfers should be next improvement?
  - Powerful capability; significant work to develop a holistic solution …
  - Or ‘carrot and stick’; provide chunked data with ‘slow FUSE virt single-file view’ vs ‘fast’ separated object view.
- Or, time to try something else entirely?
- (PS – how to avoid use of xattr requirements for checksums)

Backup

Original slides: transfer-strategy taxonomy, route map, fire-and-release, other options, TPCR sketch, summary