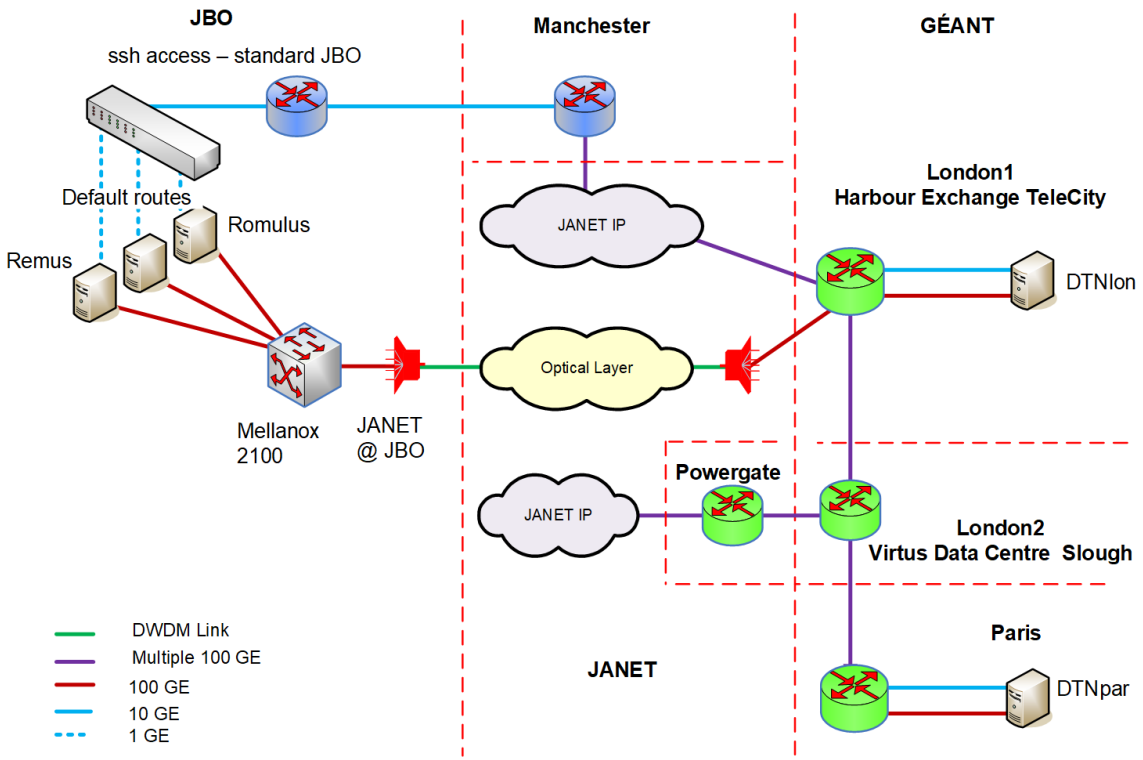
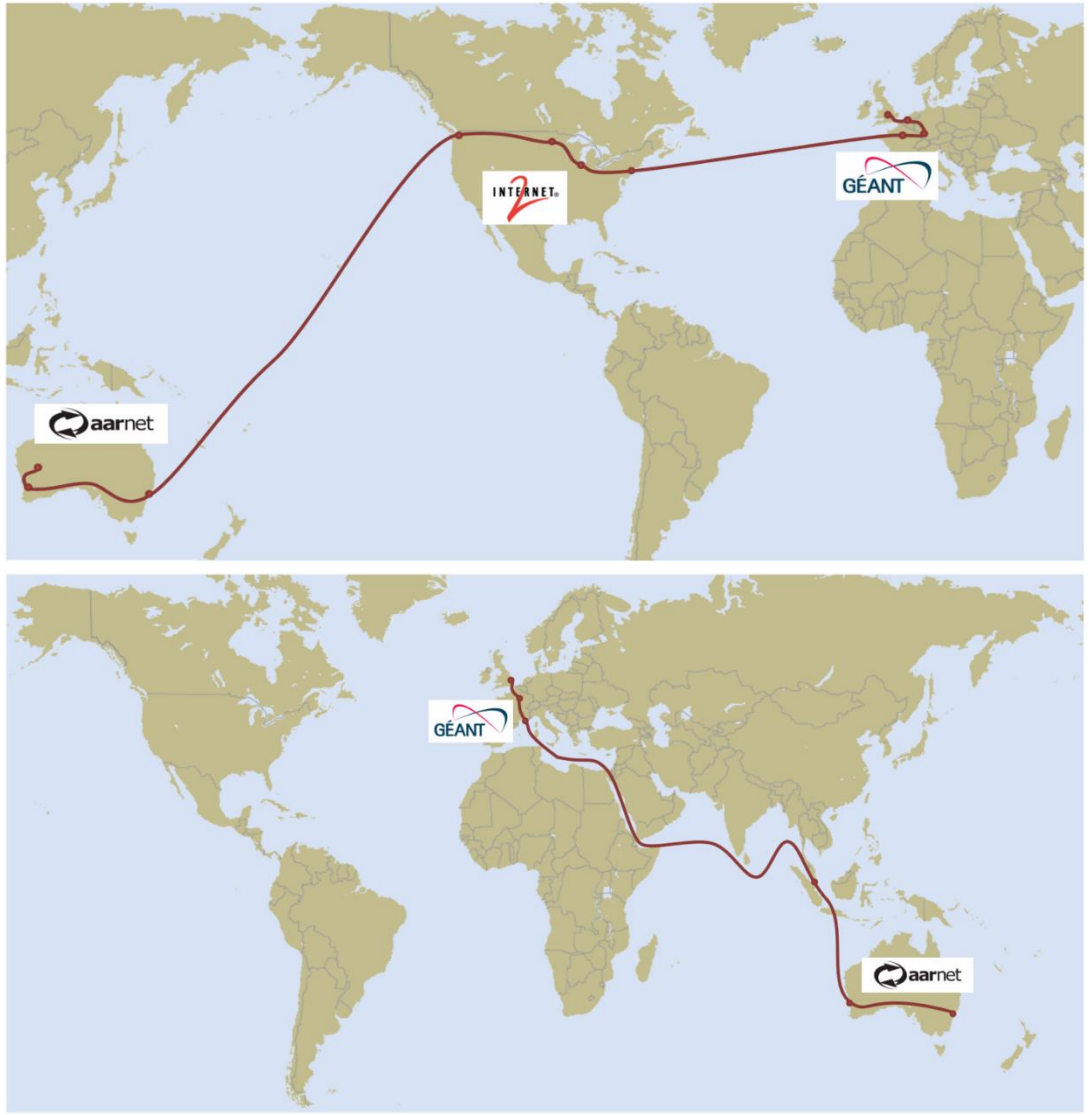
Testing the performance of data transfer protocols on long-haul high-bandwidth paths.

Richard Hughes-Jones
GÉANT Association
Tim Rayner, AARNet & Shaun Amy, CSIRO

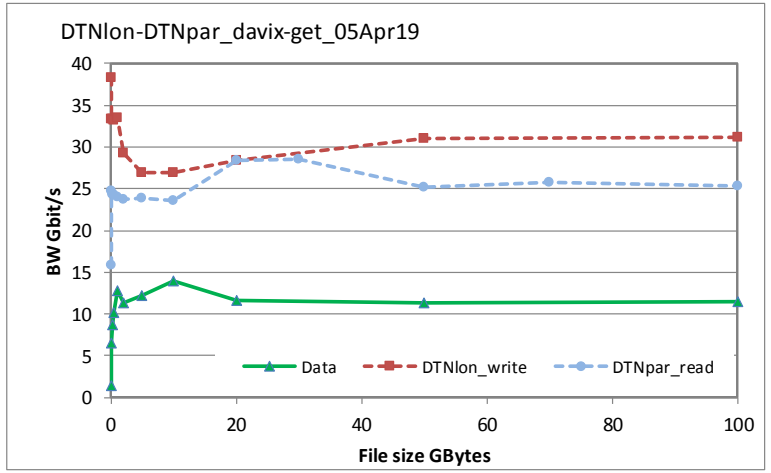
AENEAS Network Topology



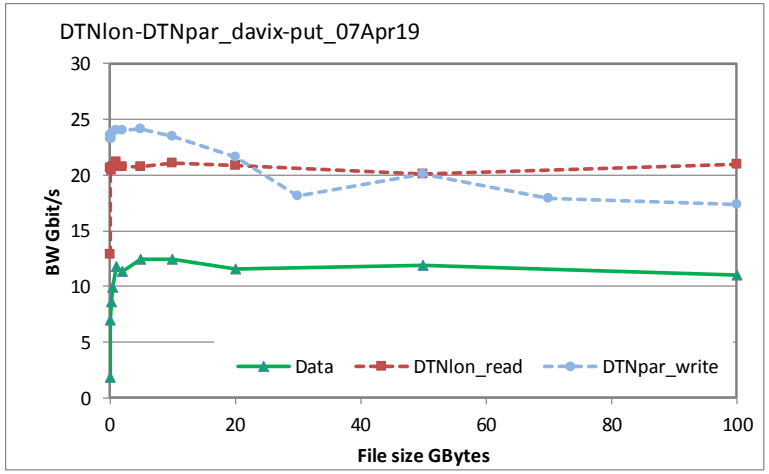


Disk to Disk Throughput vs File Size Scan davix:http

davix-get DTNlon-DTNpar



davix-put DTNlon-DTNpar



- 1 TCP flow
- Concern re low transfer speeds half disk speeds.
- Disk - memory
  Lon zfs write 30 Gbit/s
  Par xfs RAID0 read 25 Gbit/s

Instrument davix and xrdcp

- Time measurement better than 1 sec
- Set fixed size chunk reads from the network
- Measure times for “getChunk” (socket read) & “putChunk” (disk write)
- Note TCP performance parameters using tcp_info struct
- Record as time series in memory
- Option not to write a file (davix-get)

- Help from Andy Hanushevsky:
  lots of improvements in Xrootd v4.11.0
- But Tests done with v 4.8.4 !
- Using 1 TCP flow
- Did set chunk size
- So data should be OK

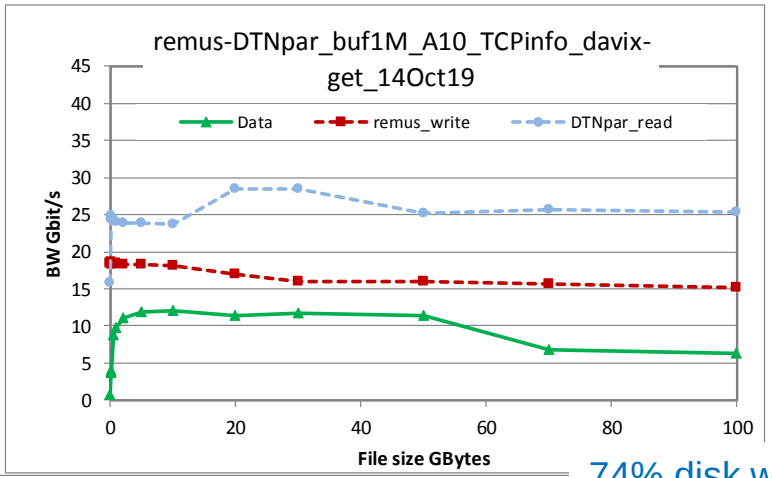
Protocol Comparaison remus ← DTNpar RTT 14.1 ms

Throughput Disk-to-Disk

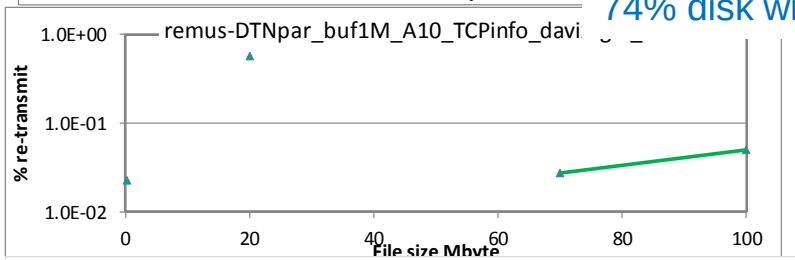
TCP re-transmits

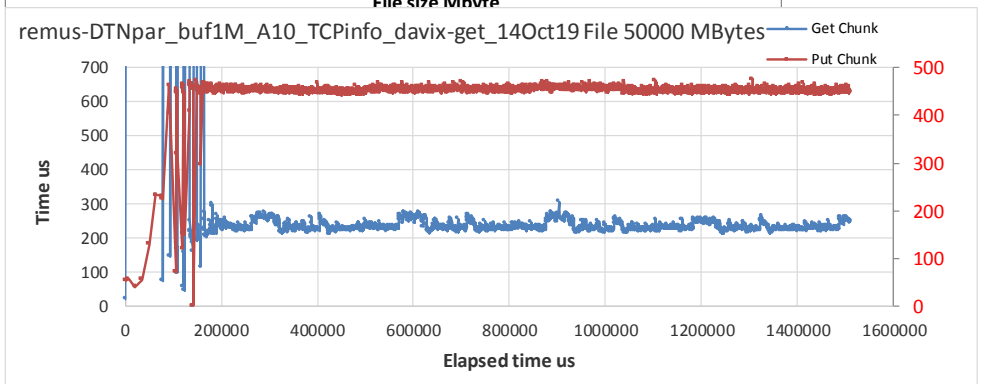
Chunk time series
Net read
Disk write

davix-get //http: 1MB buffer

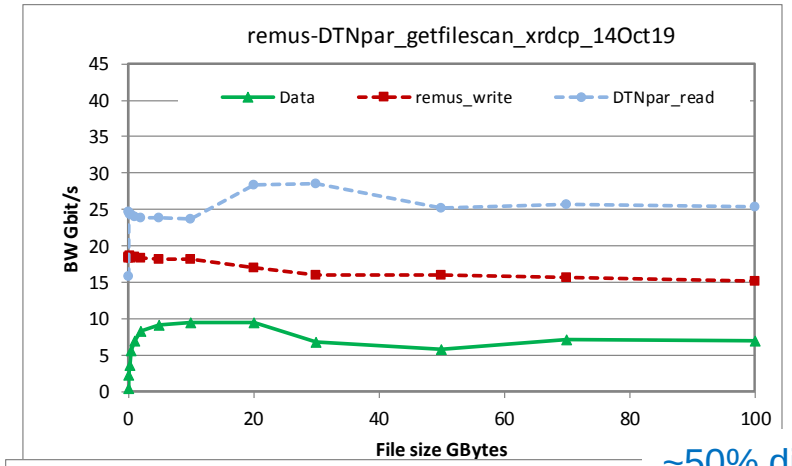


74% disk write

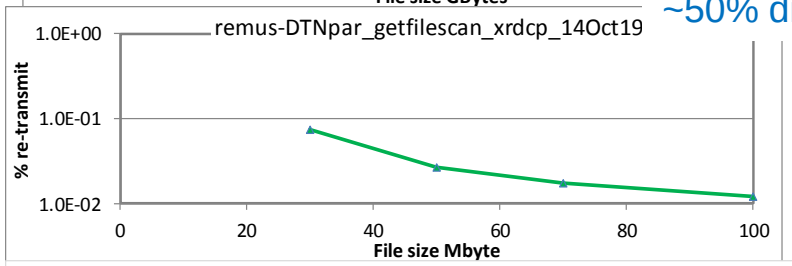


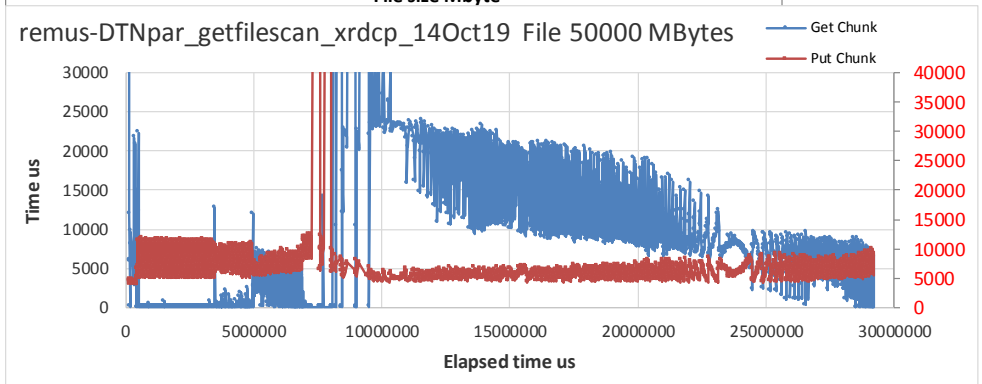


xrdcp //root: 8 MB buffer



~50% disk write

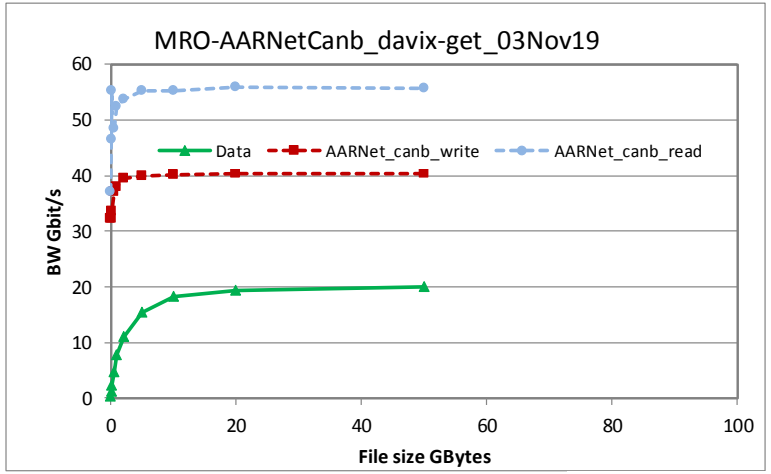




Protocol Comparaison MRO ← AARNetCanb RTT 56.5 ms

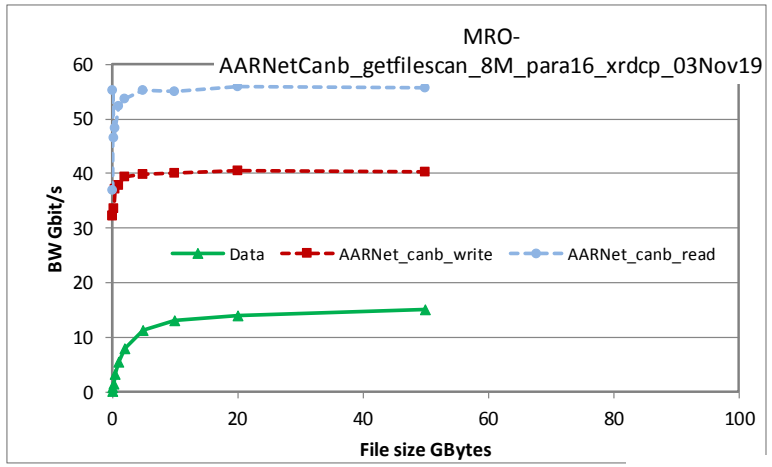
Throughput
Disk-to-Disk

davix-get //http: 1MB buffer



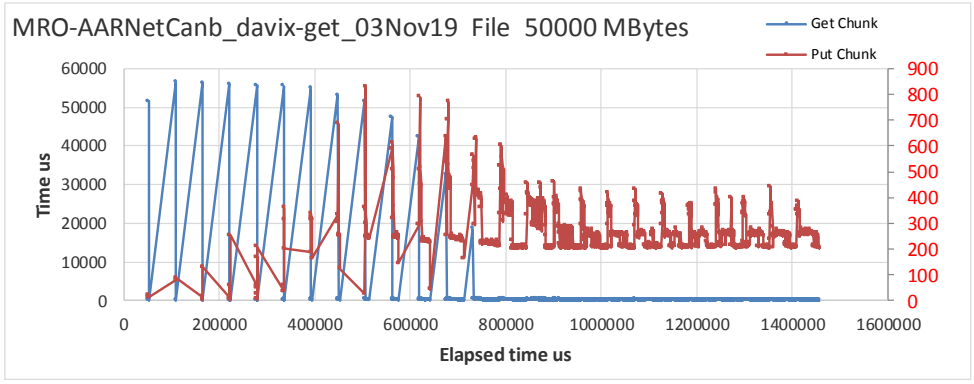
50% disk write

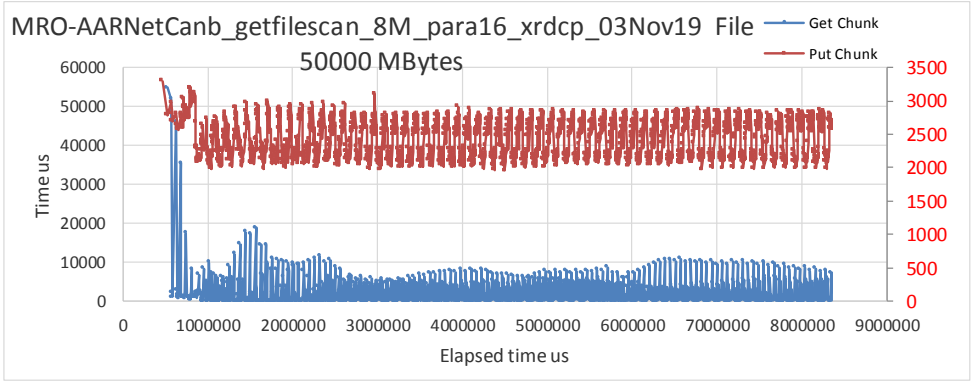
xrdcp //root: 8 MB buffer XRD // CHUNKS=16



~37% disk write

Chunk time series
Net read
Disk write



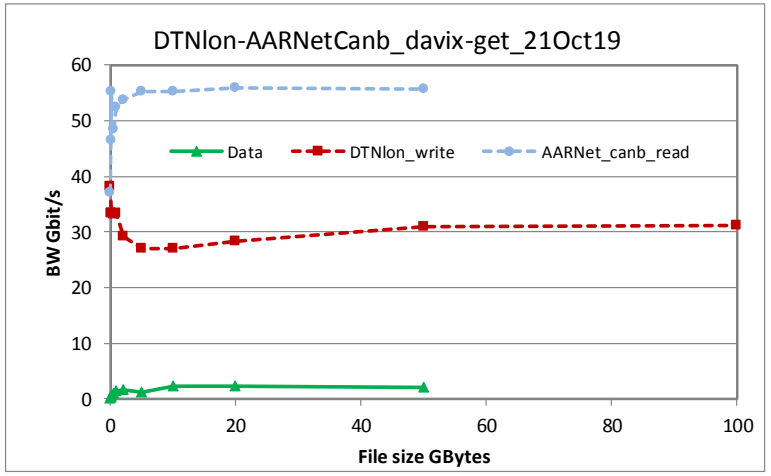


Stable after slowstart

Protocol Comparaison DTNlon ← AARNetCanb RTT 259 ms

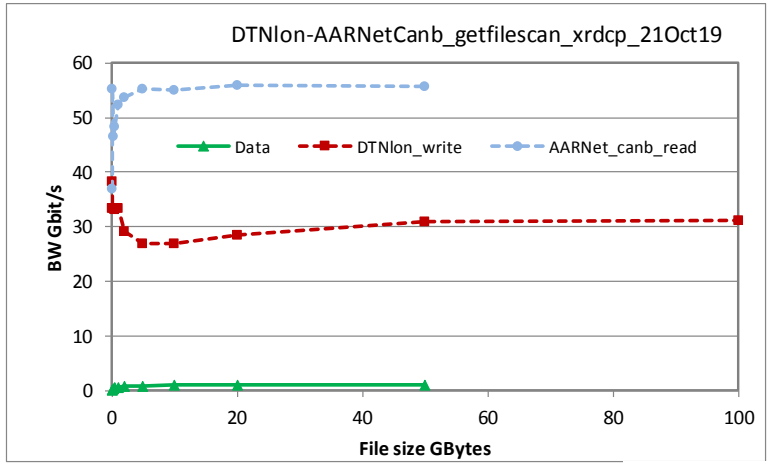
Throughput Disk-to-Disk

davix-get //http: 1MB buffer



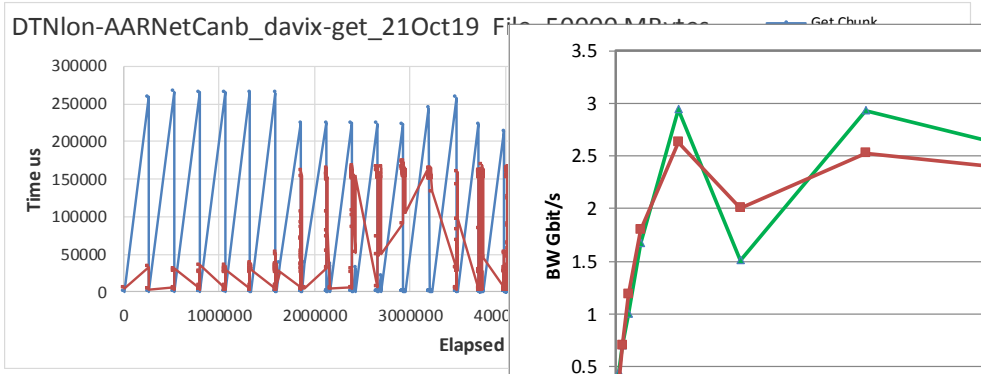
~7.4% disk write

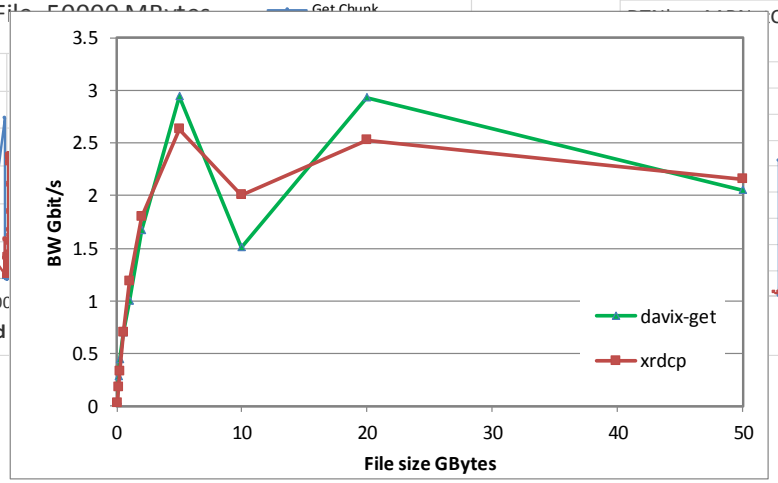
xrdcp //root: 8 MB buffer XRD // CHUNKS=16

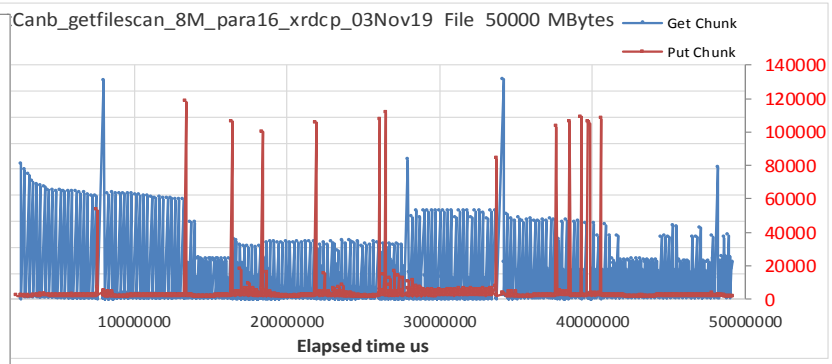


~3% disk write

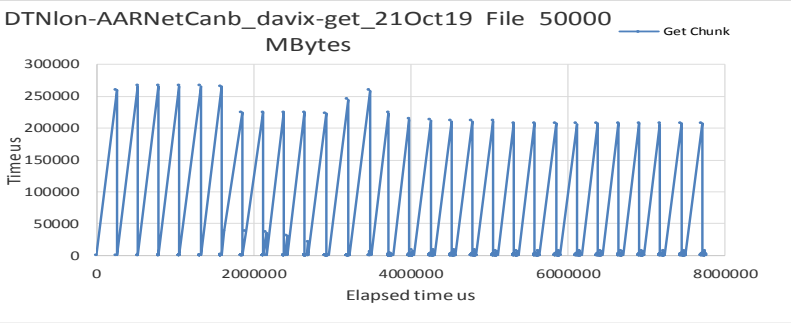
Chunk time series Net read Disk write

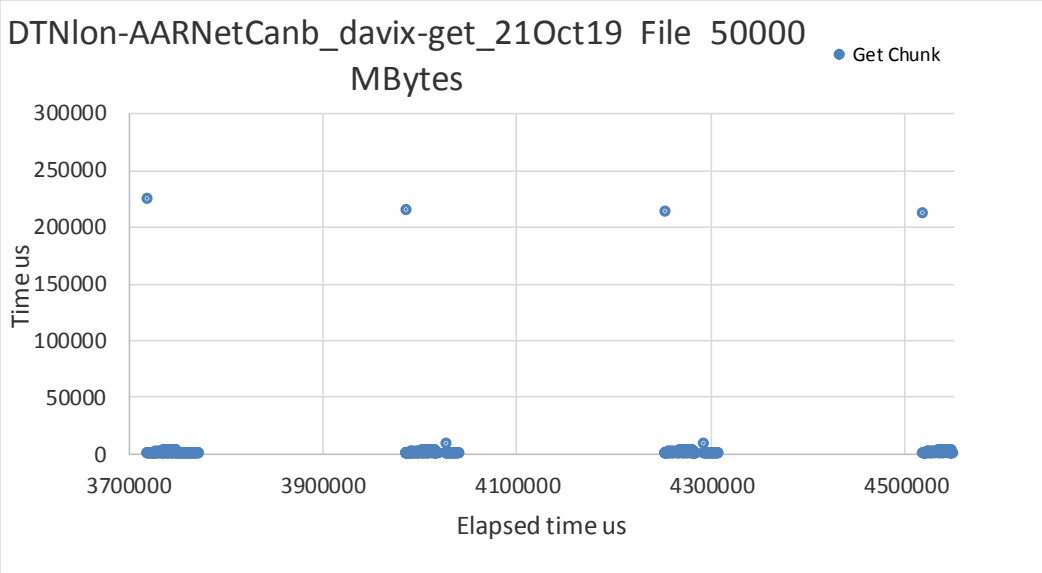






What is happening between DTNlon and AARNetCanb?





- Look at the “Get Chunk” Network read times
  - Large value = delay in reading from the network
- Would expect stability after TCP slowstart
- Slowstart ends at approx 10 * RTT 259 ms. ≈ 2.6s
- Confirmed by the receive buffer size plateau.
- Delay Bandwidth Product for 3 Gbit/s ~ 97 MBytes
- Every RTT
  Read about 100 MBytes then wait 210ms
- Why?
  - App not providing data to the TCP socket
  - TCP auto tuning is working but slower that expected over these RTT – iperf is OK
- Investigations continue

What have we learnt?

- WebDAV/http(s) and xrd protocols both work well for moving bulk data.
- A simple client loop of “Get Chunk” - “Put Chunk” clearly reduces disk-to-disk throughput.
- Use of zero-copy e.g. sendfile() on the server gives a big improvement.
- Use of multiple parallel disk accesses is a help
- TCP will send what the app gives it – keep the socket full.
- TCP auto-tuning works well at medium RTT but may be slower at large RTT.

Next steps

- Check out v 4.11.0
- Look at multiple TCP flows