Centre de Calcul de l'Institut National de Physique Nucléaire et de Physique des Particules

Source: LSST

Revisiting bulk file transfer over long distance networks

fabio hernandez

CCIN2P3

FJPPL — Japan-France workshop on computing technologies for multidisciplinary science, Lyon, Feb 15th 2017

cnrs

in2p3

BETA

- **B**ulk **E**fficient data **T**ransport for **A**stroparticle physics
- Goal: to build a prototype of an experiment-agnostic data import / export platform

  use LSST use case as a concrete problem we need to solve: 15 TB raw per night, 300 nights per year, 10 years from 2022 on

  selectively export data to both well identified and anonymous users
- Participants

  CC-IN2P3 (F. Suter, F. Hernandez), LAPP (N. Chotard, D. Boutigny)
- Duration: 3 years

State of the art and vision

- At CC-IN2P3, data import / export is currently highly dependent on the storage technology used by a given experiment

  SRM + gridFTP for LHC experiments, iRods for small groups, sftp / scp / rsync for individual users, etc.

  different levels of service, usability, performance, authentication mechanisms

- Vision: design an integrated software solution for data import / export

  similar to a cargo port of a country

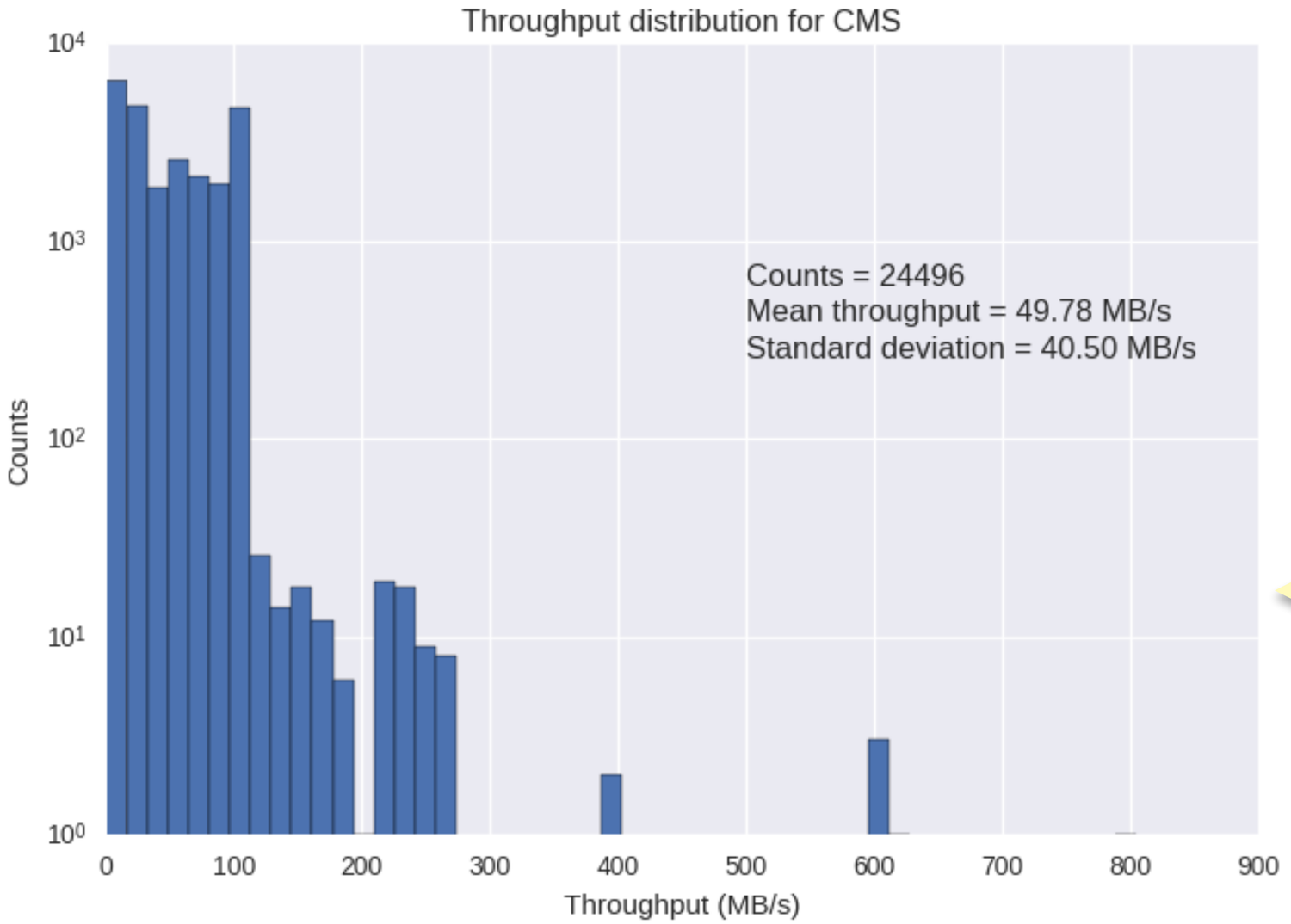
  expose REST APIs for interacting with the system, including administrative operations (e.g. stop, start, drain, restart, pause, resume, etc.) and status

  dissociate the storage platform used by the import/export platform from the permanent storage location of the data

  provide fine grained user and file authorisation mechanisms as well as mechanisms for throttling transfers

  provide mechanisms for pre- and post-processing files at emission and reception (e.g. packing/unpacking, compression/decompression, etc.)

CMS data exchange (CC-IN2P3)



12 hours period

aggregated throughput: ~400 MB/sec

import to and export from CC-IN2P3 (dCache)

Source: S. Gadrat

Guidelines

- Bottom-up approach
- First, explore the suitability of **standard protocols** for bulk data transfer

  we need a solution for the 2020-2030 period
- Protocols used to transport data for web applications include several of the features we need

  integrity & confidentiality (TLS), various standard authentication mechanisms (X509, OAuth), data transport (HTTP1 and HTTP2)

  *programmability of **both client and servers**, in multiple programming languages*

  strong industry support: future-proof
- Optimise for aggregated throughput, not latency
- Make the system operator-friendly

  emit status information, don't make assumptions on the storage platform used for emitting / receiving data, don't assume the available bandwidth is dedicated to transfers

Current status

- We are currently working on evaluating the suitability of HTTP as the transport protocol over high latency network links

  is the performance of HTTP(S) implementations comparable of that of purpose-built tools such as `gridFTP`, `FDT`, `bbcp`*?*

- What we have done so far

  developed building block tools in Go programming language

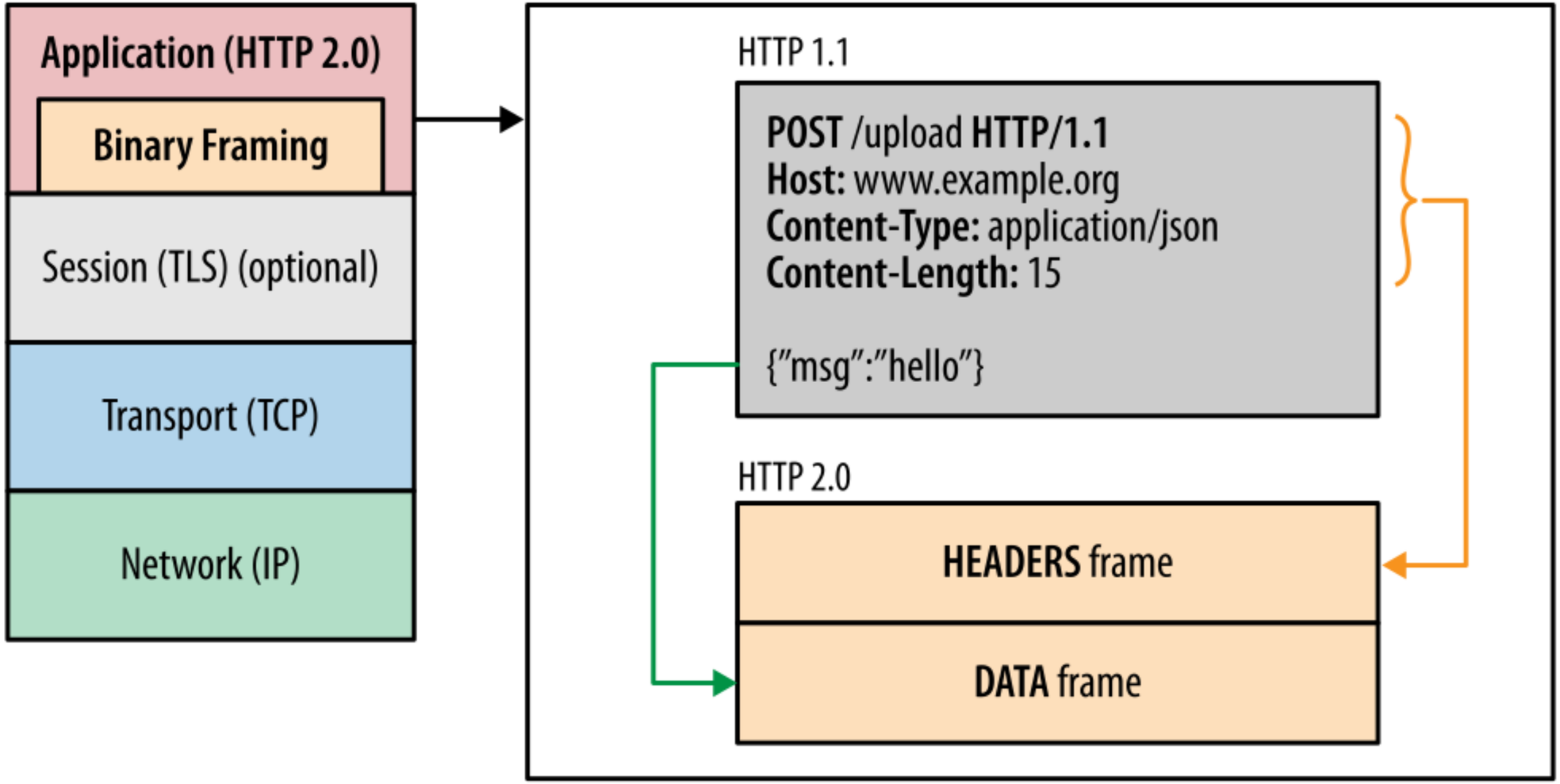
  for 1) measuring performance of memory-to-memory transfers over raw TCP and TLS and 2) measuring the time required for establishing TLS connections

  https://github.com/airnandez/netperf

  https://github.com/airnandez/tlsping

  another tool for generating HTTP2 load and measuring performance is under heavy development, not yet online

Current status (cont.)



Source: https://hpbn.co/http2

Current status (cont.)

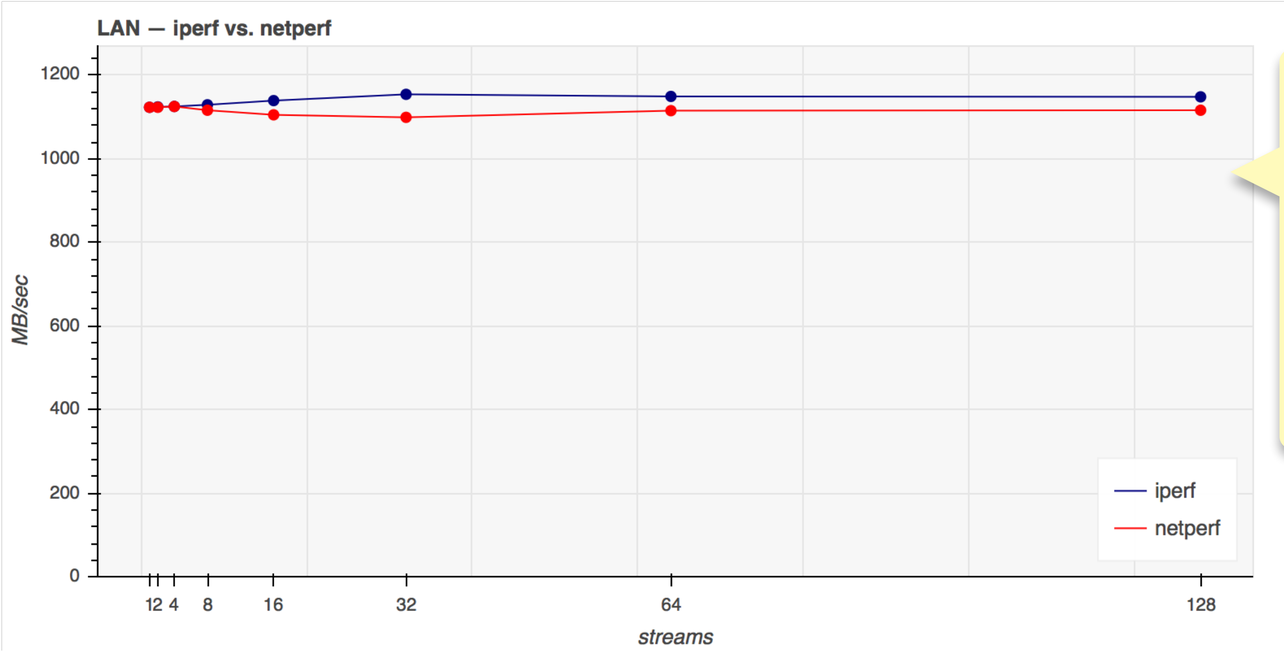
- Preliminary findings

  no noticeable penalty on throughput for network applications written in Go, as compared to tools written in lower level languages

  however, CPU consumption of Go application is higher that C/C++ but still acceptable for our use case

  in the context of a 10 Gbps LAN, we manage to exploit the available bandwidth

  overhead related to securing the communications channel using TLS is negligible provided hardware-accelerated AES is available (most Intel CPU models since ~2010)

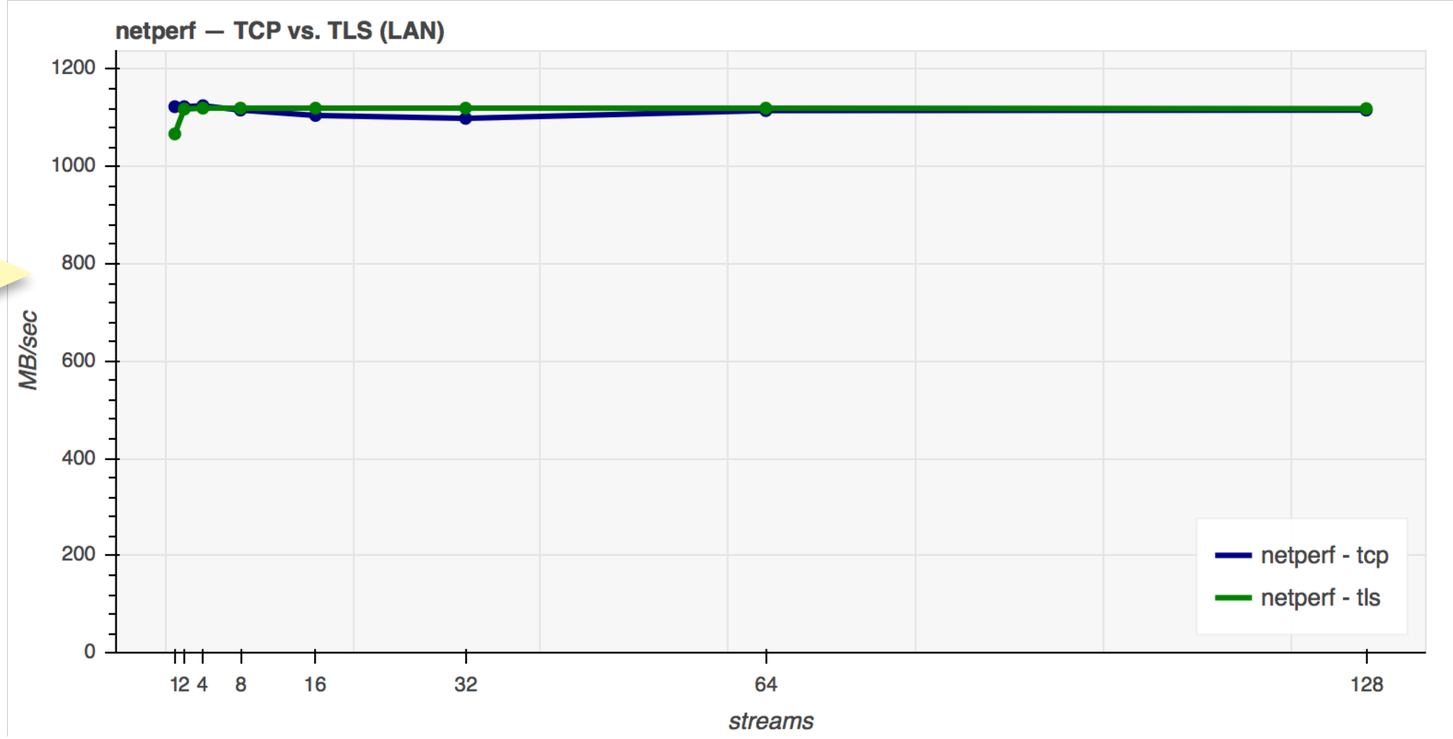


network applications written in Go can exploit the available bandwidth

(memory-to-memory transfer, local area network, 10 Gbps link)

securing the communications channel using TLS does not reduce throughput, provided we use hardware-accelerated AES

(memory-to-memory transfer, local area network, 10 Gbps link)

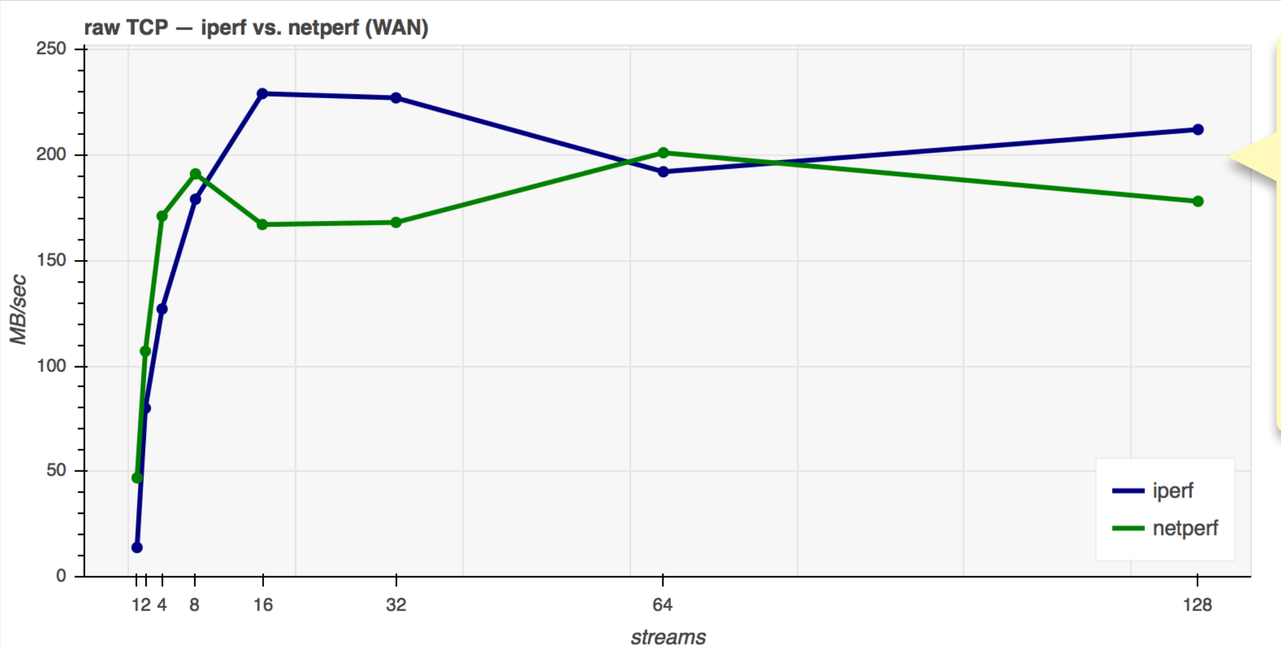


Current status (cont.)

- Preliminary findings (cont.)

  memory-to-memory transfer tests over WAN (RTT: 110 ms) show that aggregated throughput of `netperf` *is comparable to* `bbcp`

  impossible to determine the ability to exploit the available bandwidth because we don't have visibility on the activity of the end-to-end link

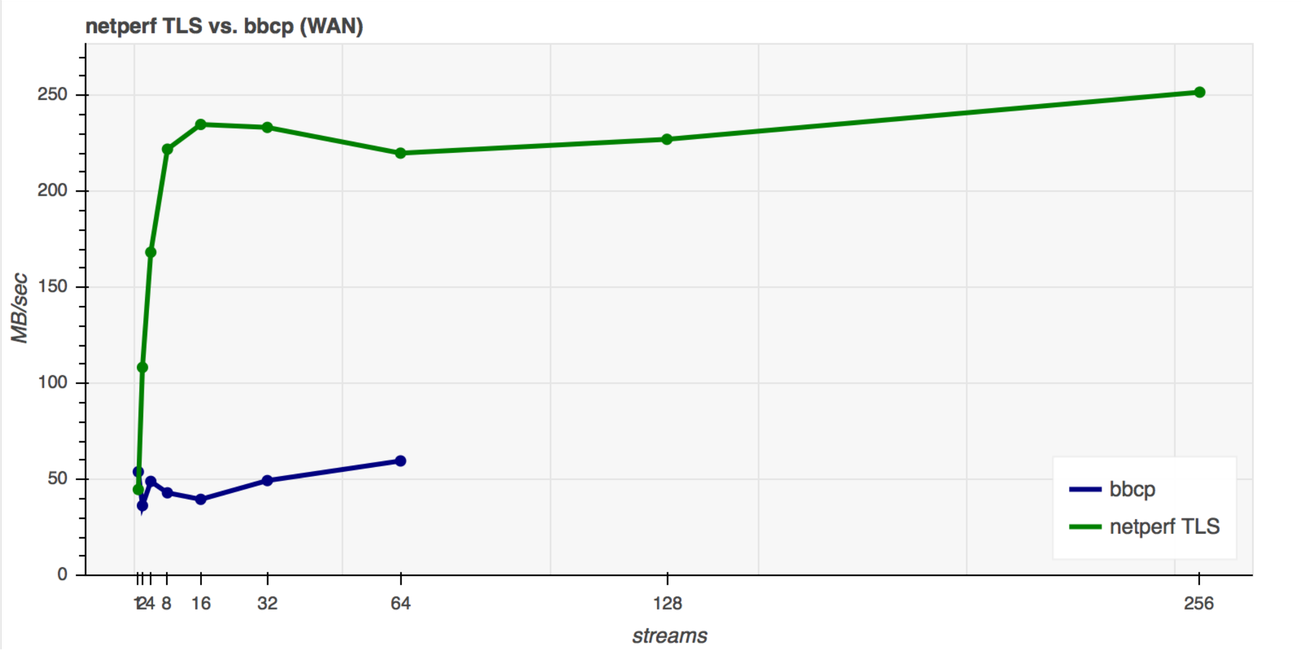


throughput of `netperf` *(Go) comparable to* `iperf3` *(C)*

(memory-to-memory transfer, wide area network, 10 Gbps shared link, RTT: 110 ms)

`netperf` *(using TLS) throughput is comparable to* `bbcp`

(memory-to-memory transfer, wide area network, 10 Gbps shared link, RTT: 110 ms)



Perspectives

- When the recently ordered testbed hardware is available, we intend to perform another campaign of tests

  both in the LAN (by simulating latency and noise) and in the WAN

- Measure the throughput using secure HTTP/2

  compare to secure HTTP/1

- Extend prototype to measure disk-to-disk transfer throughput