MLAB

@ CS&S Code for Science & Society

Bring NDT Back:
Measurement Lab Modernizes NDT Server

Chris Ritzo critzo@measurementlab.net
Matt Mathis mattmathis@measurementlab.net

@measurementlab @chrisritzo

NDT History

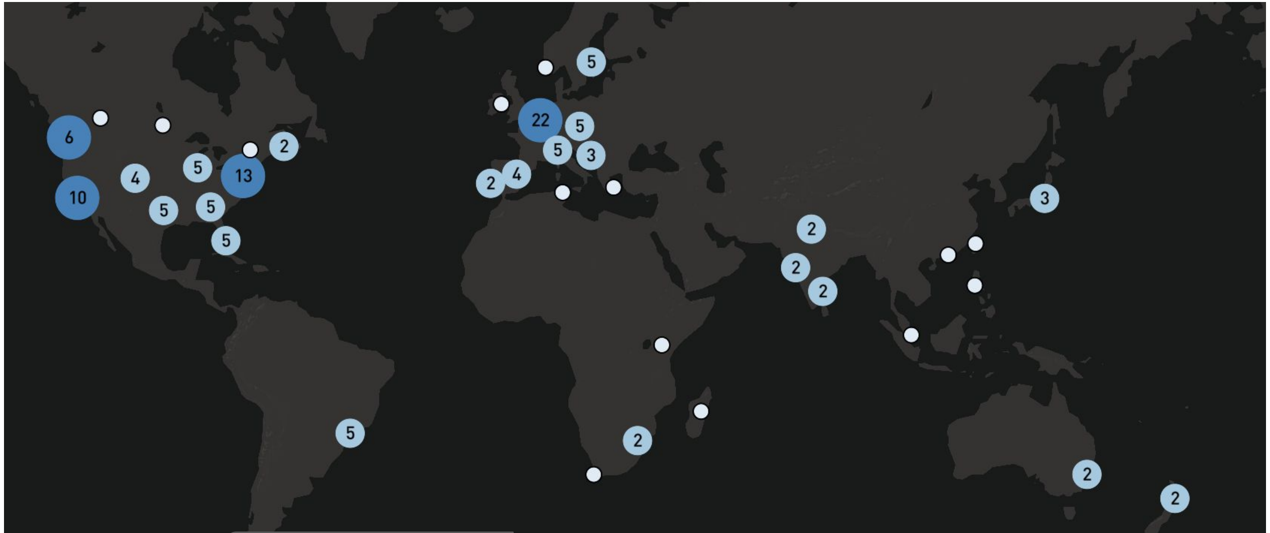
- Single stream TCP performance test developed at Internet2
- Used the *web100* kernel module for collecting tcp statistics
- Included in perfSONAR until version 4.0 release

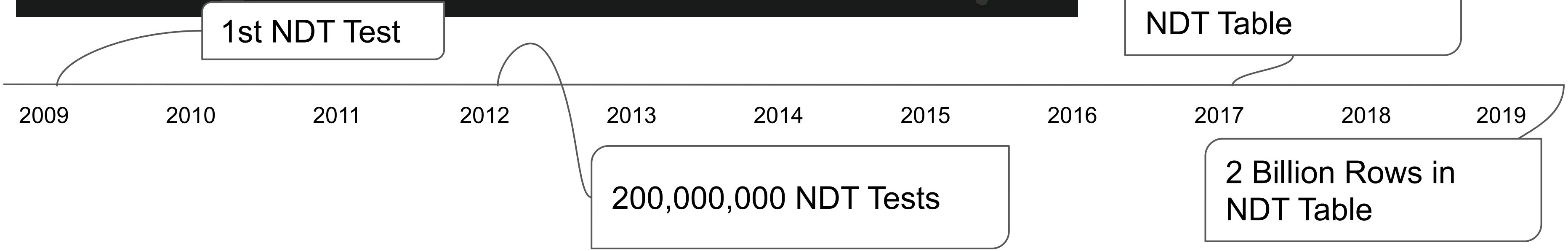
- M-Lab 1.0 continued to use NDT on our PlanetLab-based stack
  - vservers + Princeton-run api and bootserver + lots of custom tools
  - Old and floating south on an iceberg

- Required much manual patching, made ops a technical nightmare
- perfSONAR made a solid choice - chuck NDT and move on
- M-Lab was more dependent on NDT

M-Lab NDT Servers - Endpoints for measuring the public internet

MLAB @ CS&S Code for Science & Society




1st NDT Test

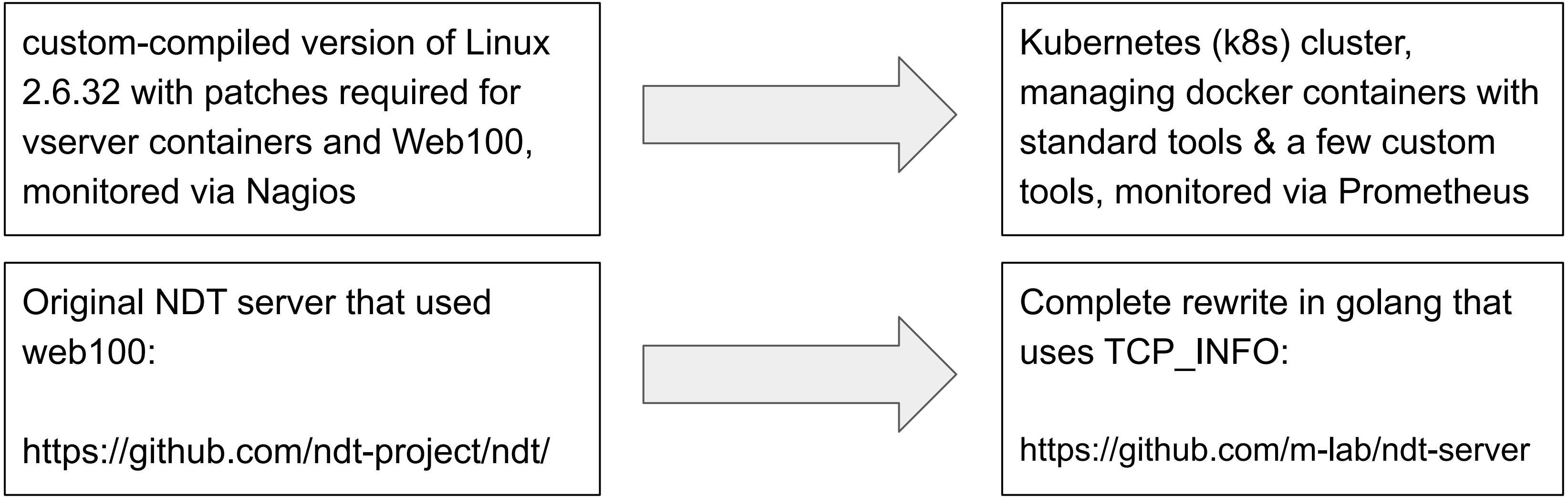
200,000,000 NDT Tests

1 Billion Rows in NDT Table

2 Billion Rows in NDT Table

Digging out of the technical debt

In 2017, the M-Lab team began work to upgrade the platform to adopt modern and flexible system administration components.



custom-compiled version of Linux 2.6.32 with patches required for vserver containers and Web100, monitored via Nagios

→ Kubernetes (k8s) cluster, managing docker containers with standard tools & a few custom tools, monitored via Prometheus

Original NDT server that used web100:

https://github.com/ndt-project/ndt/

→ Complete rewrite in golang that uses TCP_INFO:

https://github.com/m-lab/ndt-server

TL;DR: **ndt-server** can now be run using Docker, with **tcp_info**, **traceroute**, **uuid**, and **packet-header** capture “sidecar” services containers on Linux systems with kernel version >= 5.2

Original / New NDT Differences

Original NDT server

- Reno TCP Congestion Control default
- web100 for TCP statistics
- Required bidirectional ports:
  - 3001 (http)
  - 3010 (https)
  - 32768-65535 Randomly assigned ephemeral range port assigned by server for client tests

New NDT server

- ndt5 protocol
  - Backward compatible
  - Original ports supported: 3001, 3010, 32768-65535
  - Default to Cubic TCP comp.
- ndt7 protocol
  - BBR TCP, Cubic fall back
  - TLS port 443, websockets
- Supported reference clients: JavaScript, Golang, C++11
- Community clients: Android, iOS

Collaborating with perfSONAR Community

MLAB @ CS&S Code for Science & Society

- M-Lab is collaborating with perfSONAR to share ndt-server with the community
- Testing *ndt-server* with perfSONAR v4.0 CentOS release found:
  - perfSONAR
    - ships with kernel 3.10.0-957.27.2.el7.x86_64
    - Basic tests confirm that perfSONAR seems to work fine after kernel upgrade to 5.2.13-1.el7.elrepo.x86_64
    - With the kernel upgrade + docker, ndt-server can be run, but not concurrently with the perfSONAR httpd process
  - ndt-server
    - requires at least the 4.19.x LTS kernel + BBR 1.0 - (current M-Lab production)
    - BBR is still being updated, version 2.0 is a kernel module for kernels 5.2+, targeting 5.4 LTS, when it's ready
- Adding the new ndt-server into perfSONAR will be a long-term path
- But running your own ndt-server is possible now, on a separate box from perfSONAR

Test drive your own ndt-server

On a Linux machine with docker & updated kernel, run:

```
docker run --net=host measurementlab/ndt
```

Then point your browser to:

ndt5 (original proto, http)	http://localhost:3001/static/widget.html
ndt5 (original proto, TLS)	https://localhost:3010/static/widget.html
ndt7	https://localhost/static/ndt7.html

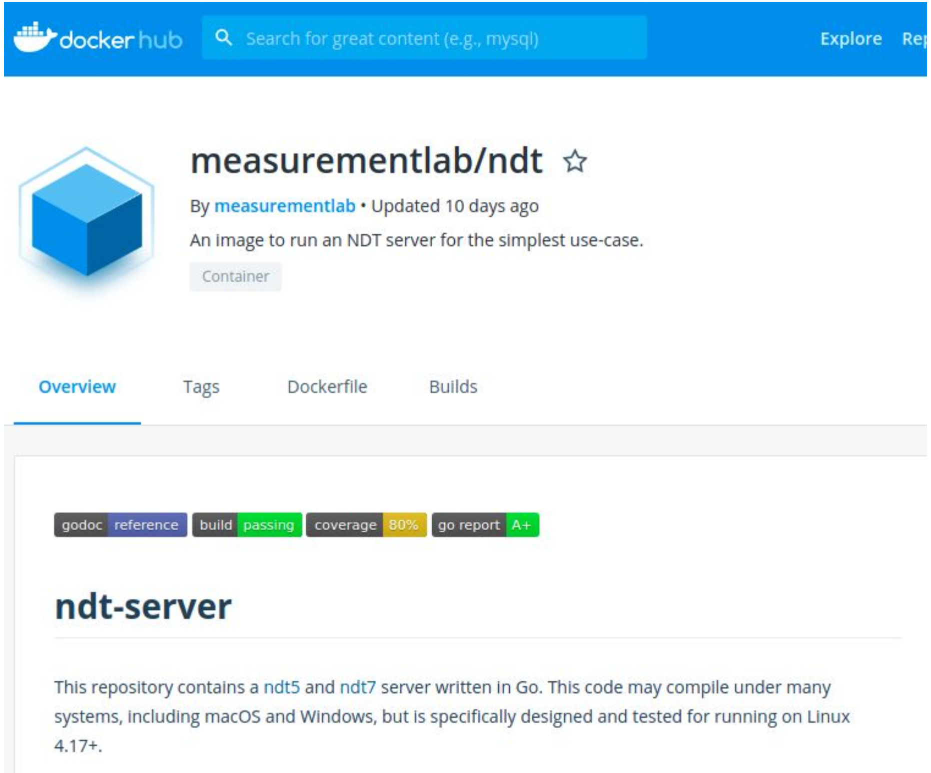
Everyone's environment will be different, and the example above is super basic.

Full Stack Demo - ndt-server

MLAB

@ CS&S Code for Science & Society

```
docker run -d -u 0
        --network=host
        --volume `pwd`/certs:/certs
        --volume `pwd`/datadir:/var/spool/ndt
        --volume `pwd`/var-local:/var/local
        --read-only --user `id -u`:`id -g`
        --cap-drop=all
measurementlab/ndt
        -cert /certs/cert.pem -key
        /certs/key.pem -datadir /var/spool/ndt
        -ndt5_addr 192.168.10.10:3001
        -ndt5_wss_addr 192.168.10.10:3010
        -ndt7_addr 192.168.10.10:4443
```



measurementlab/ndt ☆

By measurementlab • Updated 10 days ago

An image to run an NDT server for the simplest use-case.

ndt-server

This repository contains a ndt5 and ndt7 server written in Go. This code may compile under many systems, including macOS and Windows, but is specifically designed and tested for running on Linux 4.17+.

NDT's Origin revisited

- "Bulk Transport Capacity" metric as defined in [RFC 3148]
  - Test with (what was) state-of-the-art TCP
  - Instrument everything, including: web100, app performance, dispersion and full packet capture
  - Display all metrics and models derived from the metrics in the meta report
  - Enable the "user" to decide which models are important or relevant on a case-by-case basis
    - User education was an explicit goal
- But NDT fell behind in a number of ways
  - Gradual focus on raw performance and erosion of other metrics, models and understanding
  - TCP implementations are now out of date and not representative of modern stacks
  - Standard TCP (and CUBIC) is out of scale for most of the Internet
    - It has been out of scale for HPC networking/Internet 2 for nearly two decades

TCP Cubic & Reno are out-of-scale

MLAB @ CS&S Code for Science & Society

- Long standing well known problem
  - One of my focus areas for more than two decades (since 1997)
  - Previous known solutions (e.g. FAST TCP) have all failed to deploy at scale
    - All are brittle in some contexts and are not safe for unsupervised wide use
    - Most are shipped with linux and can be installed by experts as modules
- ISPs complain about NDT results
  - Want "multi stream NDT" and other changes
  - Multi-stream is really a workaround for TCP scaling issues
    - In the transport research community this is viewed as "cheating congestion control"
    - By definition this is not a "Bulk Transport Metric"

Addressing the real problem

- Core assumptions baked into Van Jacobson's landmark paper [1988]
  - VJ88 is the foundation of nearly three decades of congestion control research
  - Key principles: packet conservation and self clock
  - Unsuitable for short queue, high speed networks
    - Not enough queue space to provide a good clock for sending data
    - Self clock is intrinsically brittle in modern short queue networks
- BBR TCP finally overcomes downsides of pacing at scale
  - It is built on new core assumptions: explicitly model the network (Max_BW and min_RTT)
  - Mostly pace traffic at measured Max_BW
    - Packet transmissions are timer triggered (not by ACKs)
  - Pacing rate is dithered to update (measure) model parameters
  - See: [Cardwell et. al. "BBR: congestion based congestion control", Comm ACM 2017]

BBR Features

- You are already using it for YouTube and Google search
  - BBR solves real problems for Google
  - Several other content providers are known to be experimenting with it
    - Netflix is making good progress on a BSD port
    - Because it solves some of their problems too
- It is not done yet: the present is still a moving target
  - The version currently running on MLab (v1) has well documented bugs
    - Grossly unfair to CUBIC under some conditions (Can starve CUBIC)
    - Performs poorly over some links that batch ACKs (including WiFi with short RTTs)
  - BBRv2 is in the wings (next slide)
- MLab will re-evaluate BBR in 5.4 LTS when it propagates into CoreOS

BBRv2

- Not upstream yet
  - Easily built kernel module
- Includes a built in CUBIC compatibility mode
  - Prevents BBR from starving CUBIC
- RISK to Internet 2 community
  - CUBIC compatibility recreates some of CUBIC's lameness in future versions of BBR
  - What will happen if BBRv3 (w/o cubic compatibility) starves BBRv2?

NDT's roots, with a new twist

- The new platform uses docker
  - Think "Ultra lightweight virtual machine"
- (Nearly) fully decouples NDT from the kernel and the rest of userland
  - OS has to be new enough to run docker
  - TCP_INFO coverage and precise CC version depend on OS version
    - But NDT doesn't care (much)

Dockerized NDT

- All present and future version of Dockerized NDT will run on ANY reasonably modern Linux
  - e.g. Linux 3.10 and later
- Caveats:
  - Network and clocks have to be good enough
  - Do some minimal functional and calibration testing
  - Linux between 3.10 and 4.19 will be missing a few TCP_INFO instruments
    - But the rest of the NDT should work just fine

MLAB

@ CS&S Code for Science & Society

Bring NDT Back: Measurement Lab Modernizes NDT Server

Chris Ritzo critzo@measurementlab.net

Matt Mathis mattmathis@measurementlab.net

@measurementlab @chrisritzo