Measure Twice, Code Once

George Neville-Neil Jim Thompson

BSDCan 2015

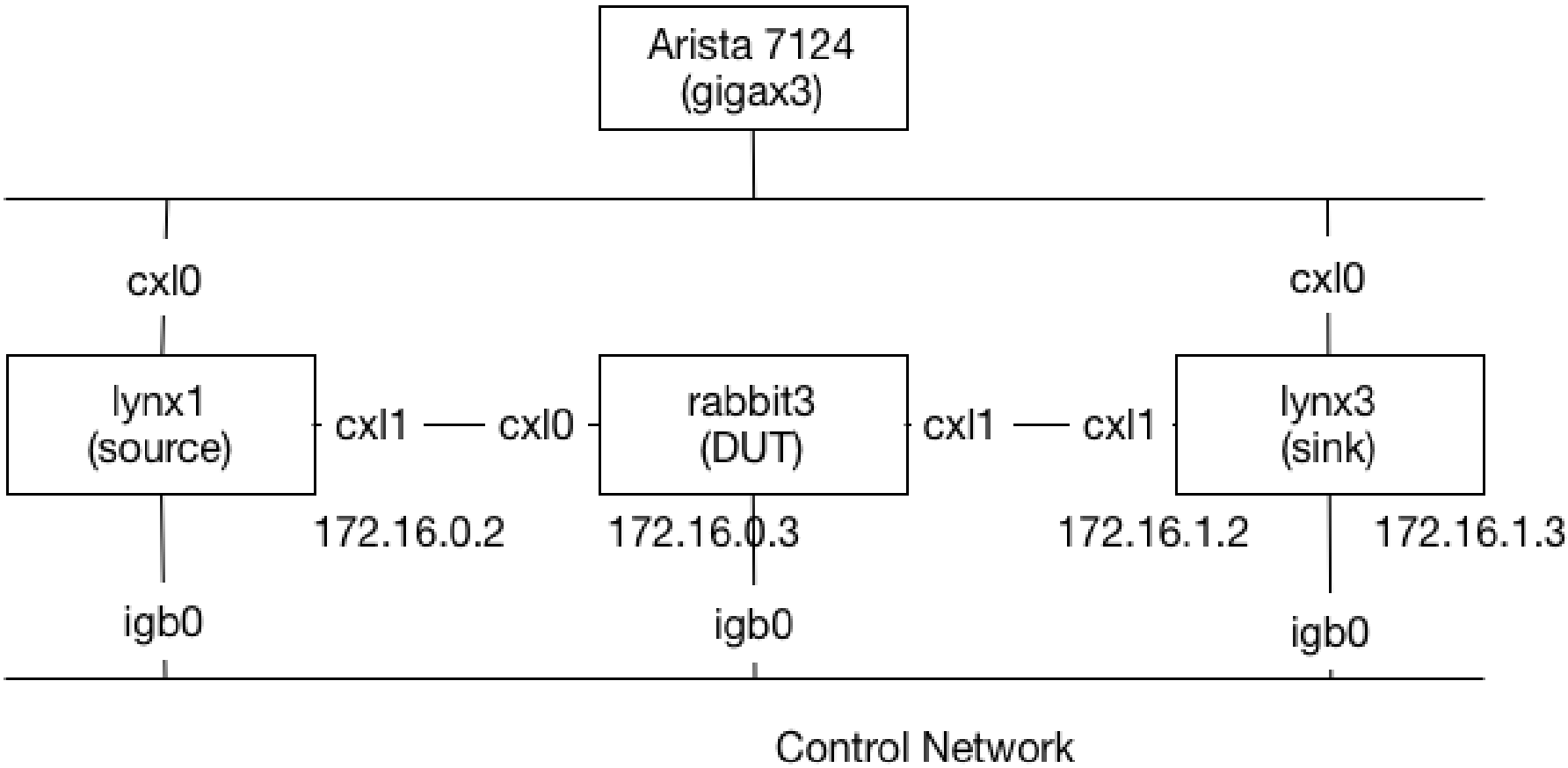
Benchmarks are Hard

- What do we measure?
- How do we measure it?
- How do we verify our measurements?
- Can our measurement be repeated?
- Can our measurement be replicated?
- Is our measurement relevant?
- How do we generate a workload?
- Does our measurement technology disturb the measurement?
  - Heisentesting

Network Benchmarks are Harder

- Asynchrony
- Best effort delivery
- Lack of open source test tools
- Control of distributed systems

Lab Setup


Arista 7124
(gigax3)
cxl0
cxl0
lynx1
(source)
cxl1
cxl0
rabbit3
(DUT)
cxl1
cxl1
lynx3
(sink)
172.16.0.2
172.16.0.3
172.16.1.2
172.16.1.3
igb0
igb0
igb0
Control Network

Hardware Used

lynx1/lynx3	dual socket, 10 core, 2.8GHz E5-2680 Xeon processors
rabbit3	single socket, four core, 3GHz E5-2637 Xeon processor
NIC	Chelsio T520, dual port, 10G NIC
Switch	Arista 7124 10G switch

Modern Hardware

- 10Gbps is 14.8 million 64 byte packets per second
- 67.5ns per packet or 200cycles at 3GHz
- Cache miss is 32ns
- Multi-core
- Multi-queue
- Lining it all up

Test Automation: Conductor

- Set of Python libraries
- *Conductor* and 1, or more, *Players*
- Four Phases

Startup	Set up system, load drivers, set routes, etc.
Run	Execute the test
Collect	Retrieve log files and output
Reset	Return system to original state

Conductor Config

```
# Master config file to run an iperf test WITHOUT PF enabled.
[Test]
trials: 1

[Clients]
# Sender
client1: source.cfg
# DUT
client2: dut.cfg
# Receiver
client3: sink.cfg
```

Player Config

```
[Master]
player: 192.168.5.81
conductor: 192.168.5.1
cmdport: 6970
resultsport: 6971

[Startup]
step1:ifconfig ix0 172.16.0.2/24
step2:ifconfig ix1 172.16.1.2/24
step3:ping -c 3 172.16.0.1
step4:ping -c 3 172.16.1.3

[Run]
step1:echo "running"
step2:pmcstat -O /mnt/memdisk/pktgen-instruction-retired.pmc -S instruction-retired -l 25

[Collect]
step1:echo "collecting"
step2:mkdir /tmp/results
step3:cp -f /mnt/memdisk/pktgen-instruction-retired.pmc /tmp/results/
step4:pmcstat -R /tmp/results/pktgen-instruction-retired.pmc -G \
          /tmp/results/pktgen-instruction-retired.graph
step5:pmcstat -R /tmp/results/pktgen-instruction-retired.pmc -D /tm/results -g
step6:pmcannotate /tmp/results/pktgen-instruction-retired.pmc \
            /boot/kernel/kernel > /tmp/results/pktgen-instruction-retired.ann

[Reset]
step1:echo "system reset: goodbye"
```

Host to Host Baseline Measurement

iperf TCP based test

netperf Packet based test using `netmap(4)`

Baseline TCP Measurement

```
0.00-1.00   sec 1.09 GBytes 9.41 Gbits/sec
1.00-2.00   sec 1.10 GBytes 9.41 Gbits/sec
2.00-3.00   sec 1.10 GBytes 9.41 Gbits/sec
3.00-4.00   sec 1.10 GBytes 9.41 Gbits/sec
4.00-5.00   sec 1.10 GBytes 9.41 Gbits/sec
5.00-6.00   sec 1.10 GBytes 9.42 Gbits/sec
6.00-7.00   sec 1.10 GBytes 9.41 Gbits/sec
7.00-8.00   sec 1.10 GBytes 9.41 Gbits/sec
8.00-9.00   sec 1.10 GBytes 9.41 Gbits/sec
9.00-10.00 sec 1.10 GBytes 9.41 Gbits/sec
```

Baseline pkt-gen Measurement

- Source

```
827.257743 main_thread [1512] 14697768 pps
828.259812 main_thread [1512] 14668997 pps
829.261742 main_thread [1512] 14695277 pps
830.263743 main_thread [1512] 14685547 pps
```

- Sink

```
866.466039 main_thread [1512] 11943109 pps
867.468024 main_thread [1512] 11946111 pps
868.469126 main_thread [1512] 11942020 pps
869.471027 main_thread [1512] 11939957 pps
```

Baseline Discussion

- TCP uses full sized packets
- pkt-gen uses minimum sized (64 byte) packets
- The DUT cannot quite keep up

IPSec and its Algorithms

- Encryption is computationally expensive
- Offloaded co-processors
- On chip instructions `AESNI`

Measurement Methods

- Two host setup
- iperf3 using TCP
- Conductor sets up the tests
- 10 rounds of 10 seconds each

Baseline

- Using NULL methods
- No authentication or encryption
- No TCP offload on the NIC cards

Min	Max	Median	Avg	Stddev
2.24	2.48	2.25	2.2844444	0.079390036

Authentication

- HMAC-SHA1 authentication
- Transport mode
- No encryption

Min	Max	Median	Avg	Stddev
615	632	628	623.3	7.9867947

AES-GCM

- Tunnel Mode
- Both encryption and authentication
- Results with and without hardware support

HW Suppt	Min	Max	Median	Avg	Stddev
N	273	280	276	276.55	2.12
Y	1220	1300	1270	1268.88	0.023

Overall Picture

Algorithm	Min	Max	Median	Avg	Stddev
NULL	2240	2480	2250	2284.44	0.079
HMAC-SHA1	615	632	628	623.3	7.98
AES-GCM Soft	273	280	276	276.55	2.12
AES-GCM Hard	1220	1300	1270	1268.88	0.023

- Want to find out *Why?*
- Come to VBSDCon 2015.

Firewall Comparison

- pfSense 2.2 (FreeBSD 10.1)
- OpenBSD 5.6
- FreeBSD HEAD (GENERIC-NODEBUG)
- Linux iptables

- Firewall Rules are given in the Paper

Single Core w/o Filtering

OS Version	PPS	StdDev
pfSense 2.2	494,224	1944
OpenBSD,5.6	360,147	1162
FreeBSD 11-CURRENT	249,464	498
CentOS 7	198,239	172

Single Core with Filtering

OS Version	PPS	StdDev
pfSense 2.2	228,558	1440
OpenBSD 5.6	187,523	78
CentOS 7	139,797	95
FreeBSD 11-CURRENT	131,795	229

Multicore w/o Filtering

OS Version	Multi-Core	Single Core	Speedup
CentOS 7	945,807	198,239	4.7x
pfSense 2.2	920,415	494,224	1.8x
FreeBSD 11-CURRENT	684,721	249,464	2.4x
OpenBSD 5.6	361,253	360,147	N/A

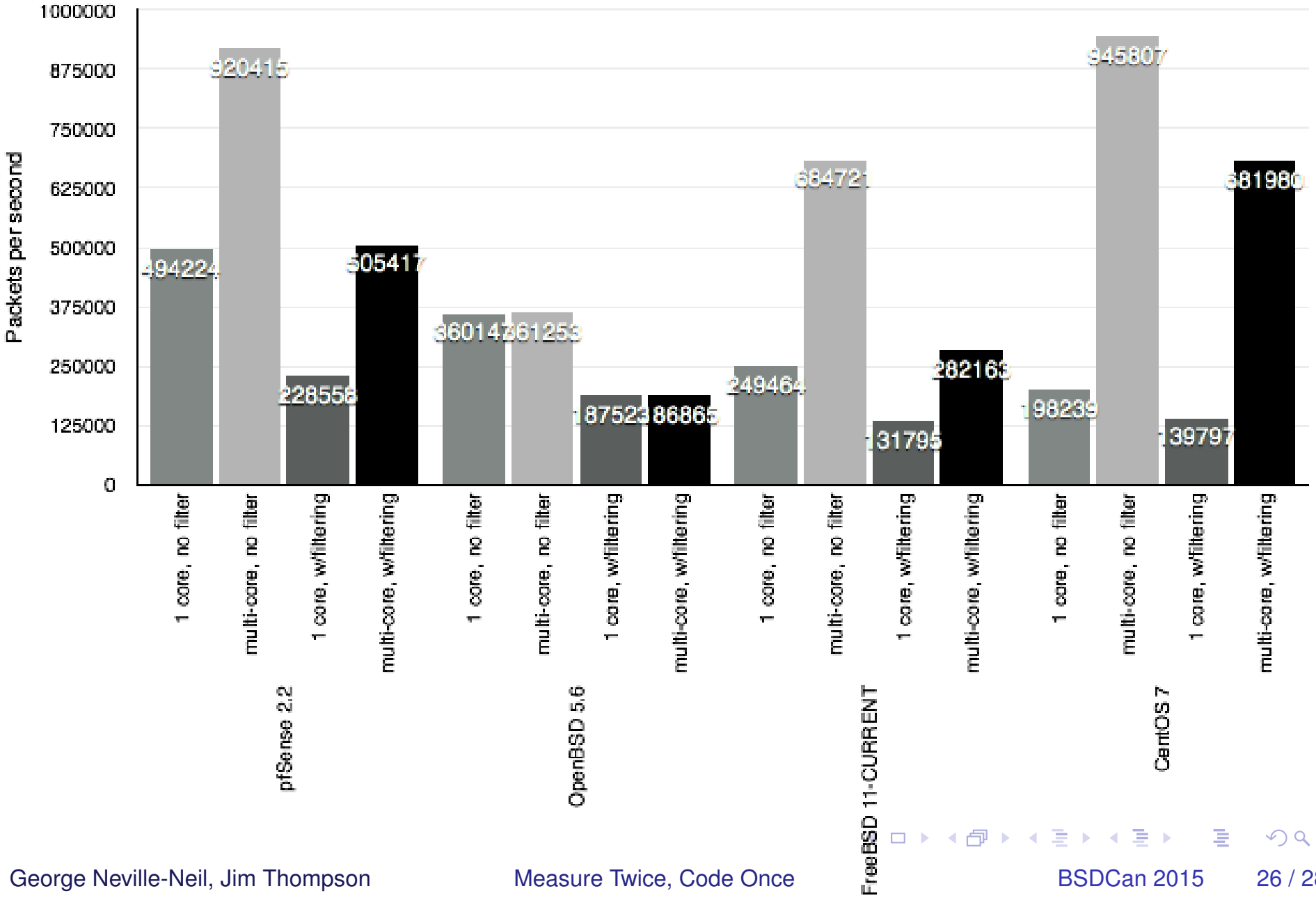
Multicore with Filtering

OS Version	Multi-Core	Single Core	Speedup
CentOS 7	681,980	139,797	4.8x
pfSense 2.2	505,417	228,558	2.2x
FreeBSD 11-CURRENT	282,163	131,795	2.1x
OpenBSD 5.6	186,865	187,523	N/A

Discussion

- Answers and more questions
- Mutli-core Matters
- Fast Multi-core matters even more
- Why is iptables the fastest?
- Why does FreeBSD lag pfSense, which is based on FreeBSD?

The Full Picture



An Ongoing Longitudinal Study

- Conducted continuously
- Reported several times per year
- Covering more subsystems
- Next Update: VBSDCon 2015

Where to get it all

Netperf `http://github.com/gvnn3/netperf`

- Includes scripts and results

Conductor `http://github.com/gvnn3/conductor`

- The test framework

pfSense `http://www.pfsense.org`

FreeBSD `http://www.freebsd.org`

Raj Jain *The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling*