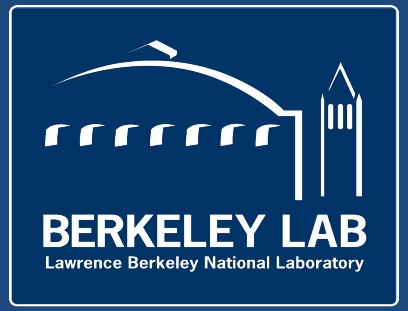


U.S. DEPARTMENT OF
ENERGY

UNIVERSITY OF
CALIFORNIA

BERKELEY LAB
LAWRENCE BERKELEY NATIONAL LABORATORY

U.S. DEPARTMENT OF ENERGY

Fast IOC searching with a columnar database (ClickHouse)

Michael Smitasin

Cyber Security Engineer

security@lbl.gov

2019-04-17

Quick intro

- First NSM Meeting!
- Started cyber security in May 2018
- LBNL since Aug 2013
- Public Sector / R&E since 2008
- Background was previously network engineering
  - some sysadmin, vm/storage admin, desktop support prior

The Problem

- Determine if there are "hits" on IOCs as quickly as possible
- Reduce Mean-Time-To-Innocence
- The Prompt:
  - "Tell me if this IP address was seen in the last 90 days in less than 1 sec"
    - (or last 3 years in ~a few secs)

The traditional solutions

- Recursive grep
  - zfgrep -r 192.0.2.49 /data-BRO/bro-NFS/WIRED/logs/2019/*/*/conn.log*
  - exec time: ~28 hours for ~44.6 billion rows (~90 days)

- GNU parallel
  - find /data-BRO/bro-NFS/WIRED/logs/ -type f -name "conn.log*" | parallel -j90 --eta 'zfgrep 192.0.2.49 {}'
  - exec time: ~19 mins for ~44.6 billion rows (~90 days)

The traditional solutions

- Row-oriented databases
  - SELECT ts, id.orig_h, id.resp_h WHERE id.orig_h = '192.0.2.70' or id_resp_h = '192.0.2.70'
  - exec time: I didn't actually test it

	ts	uid	id.orig_h	id.orig_p	id.resp_h	id.resp_p	proto	conn	l_orig	l_resp	missed	hist
1	1542095993.756999	CJ5v1C4k7aYxGIqVe2	192.0.2.10	44148	192.0.2.247	20128	tcp	S0	F	T	0	S
2	1542095993.766248	CMFxyc3oa9qzV0P8Je	192.0.2.27	31458	192.0.2.67	199	tcp	S0	F	T	0	S
3	1542095993.766739	CsIQAF8lYtdz6UIzb	192.0.2.27	31458	192.0.2.70	554	tcp	S0	F	T	0	S
4	1542095993.769726	CEvs5o4XX361jjczVl	192.0.2.220	43180	192.0.2.255	23	tcp	S0	F	T	0	S
5	1542095993.773877	C2McX695kczyZ520vc	192.0.2.11	56428	192.0.2.141	44974	tcp	S0	F	T	0	S

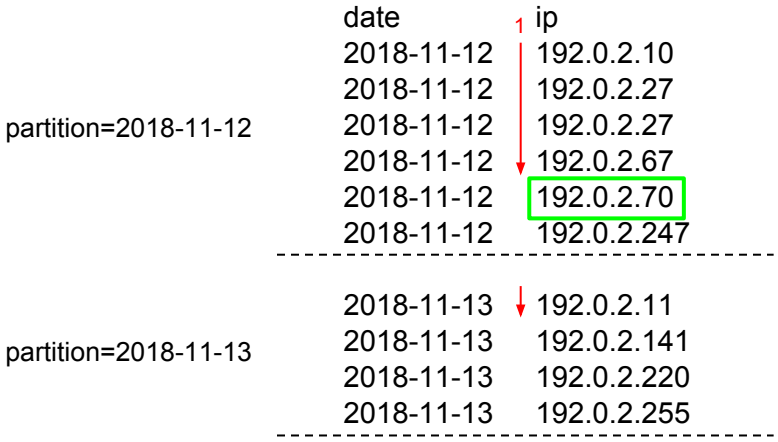
The new solution

- Columnar database (ClickHouse)
  - SELECT ts, id.orig_h, id.resp_h WHERE id.orig_h = '192.0.2.70' or id_resp_h = '192.0.2.70'
  - exec time: ~21 secs for ~44.6 billion rows (~90 days)

ts	uid	id.orig_h	id.orig_p	id.resp_h	id.resp_p	proto	conn	l_orig	l_resp	missed	hist
1542095993.756999	CJ5v1C4k7aYxGIqVe2	192.0.2.10	44148	192.0.2.247	20128	tcp	S0	F	T	0	S
1542095993.766248	CMFxyc3oa9qzV0P8Je	192.0.2.27	31458	192.0.2.67	199	tcp	S0	F	T	0	S
1542095993.766739	CsIQAF8lYtdz6UIzb	192.0.2.27	31458	192.0.2.70	554	tcp	S0	F	T	0	S
1542095993.769726	CEvs5o4XX361jjczVl	192.0.2.220	43180	192.0.2.255	23	tcp	S0	F	T	0	S
1542095993.773877	C2McX695kczyZ520vc	192.0.2.11	56428	192.0.2.141	44974	tcp	S0	F	T	0	S

The new solution: even faster

- Columnar database (ClickHouse)
  - SELECT date, ip WHERE ip = '192.0.2.70'
  - exec time: ~1sec for ~101.2 billion rows (~90 days)



- IPs are **sorted** within each partition (day)
- Each conn has 2x rows (1 for each IP)
- We can't tell if the IP was src or dst
  - (see page: Workflow)

ClickHouse 

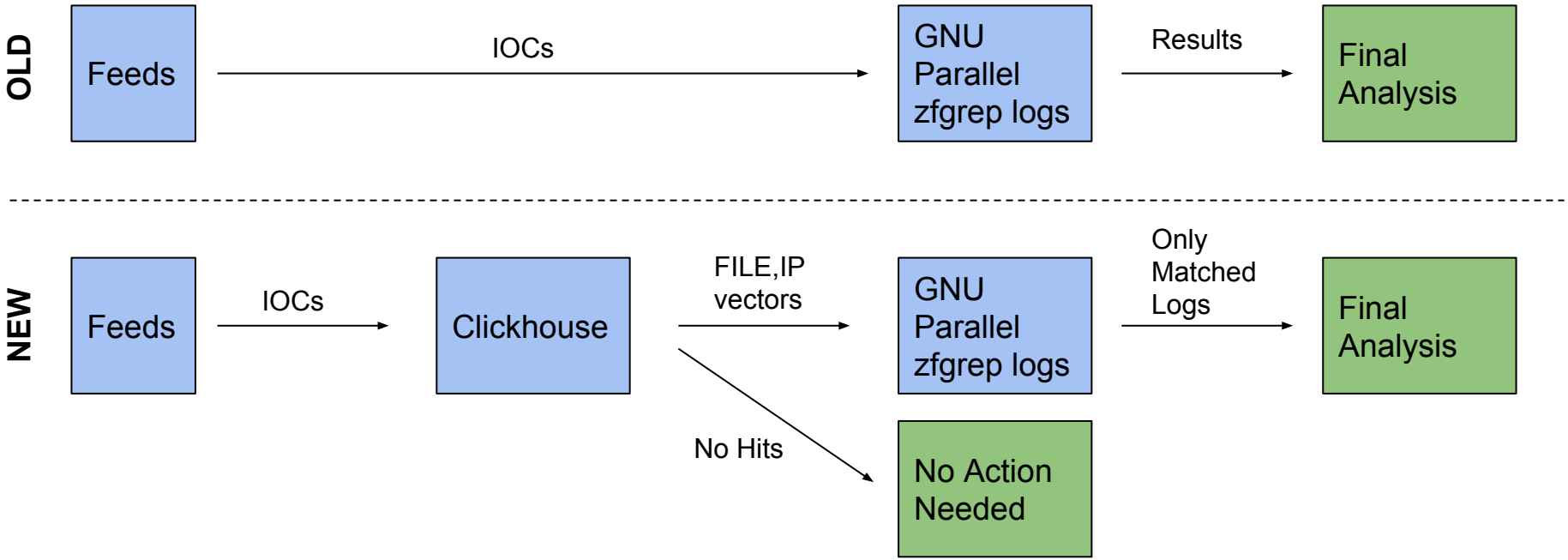
- Free, Open Source
- Off-the-Shelf
- Easy to install
  - Debian/Ubuntu easiest, official
  - RHEL/CentOS/Fedora doable, used to be unofficial, less tested
  - Maybe BSD ports, could also compile from source
  - Docker containers available
- Developed/used by Yandex (.RU equivalent of Google)

Examples

- 90 day searches:
  - 1.1.1.1 ~0.733s
  - 8.8.8.8 ~1.746s
  - 128.15.* ~0.658s

- 3 year searches:
  - 1.1.1.1 ~1.698s
  - 8.8.8.8 ~5.742s
  - 192.0.2.242 ~1.606s (negative result)

Workflow


OLD
Feeds
IOCs
GNU Parallel zfgrep logs
Results
Final Analysis
NEW
Feeds
IOCs
Clickhouse
FILE,IP vectors
GNU Parallel zfgrep logs
Only Matched Logs
Final Analysis
No Hits
No Action Needed

Just an indexer?

- Primary mission is indexer
  - Negative response ASAP
  - Positive responses -> **targeted**, rich searches
  - Could optimize further (uniq IPs, exclude local IPs) but...

- Secondary mission
  - Connection counts = cheap, interesting
  - ClickHouse = fast analytics:
    - Top10 (orig_cc, proto, resp_p) where conn_state=S0
    - Top10 (5-tuple) where conn_state=SF and pkts<2
    - Top10 non-US orig_cc services by bytes

Integration

- Query script… 'clicksearch-ip-fast.sh -C 8.8.4.4 -90'
  - Search type, IP, days
  - Wrapper for queries over SSH
  - Ease-of-use
  - Portability

- Ingest script… 'PUSH:Clickhouse-zeek'
  - Nightly cron
  - SCPs log files
  - Formats and filters data

Hardware

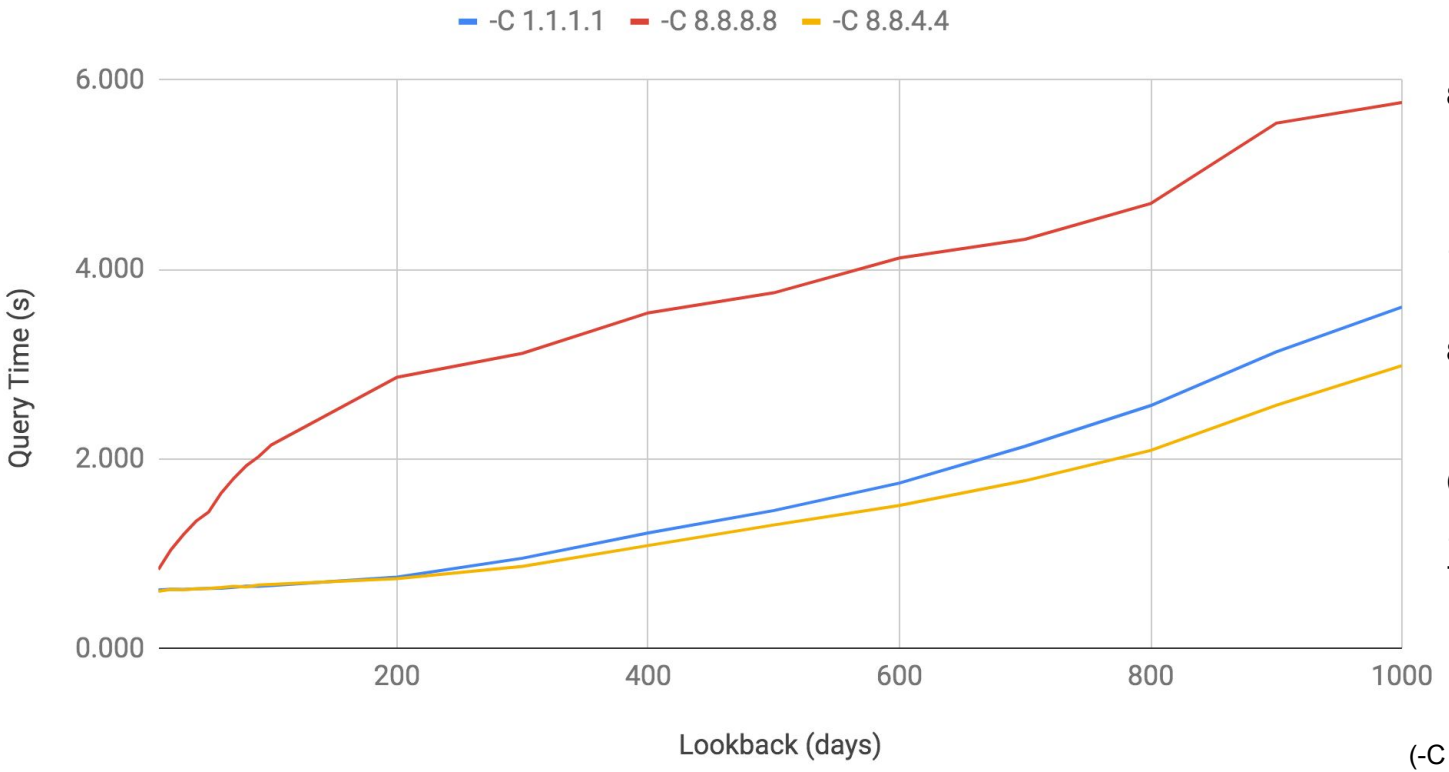
- Proof-of-Concept (now)
  - Supermicro X10DRi
  - Intel E5-2650 v4 @ 2.20GHz
  - 48 cores (HT)
  - 128GB RAM
  - 2x 240GB SSDs in RAID 1
- Production (soon)
  - Supermicro X10QRH+
  - Intel E5-4669 v4 @ 2.20GHz
  - 176 cores (HT)
  - 512GB RAM
  - 18x 2TB SSDs in RAID 10

- More cores, lower clock speed is better than fewer cores, higher clock speed
- LBL currently uses 1 box but ClickHouse supports HA, clustering/sharding
- I haven't done any special tuning for ClickHouse (yet)

PoC Lessons Learned

- Schemas & data formatting… picky
- Partitioning for "ring buffer" (default month… can do day)
  - Have to drop a partition, can't just drop rows by date
- Pre-sorting = speed!
  - 90 day search = ~1s sorted, ~20-30s unsorted
- Can ingest 3 years, only SELECT 90 days, still fast
- SELECTing more columns affects search time!

Sparse (1.1.1.1, 8.8.4.4) vs. Dense (8.8.8.8) Results



8.8.8.8
Avg: 1.12M / day
Total: 1.22B

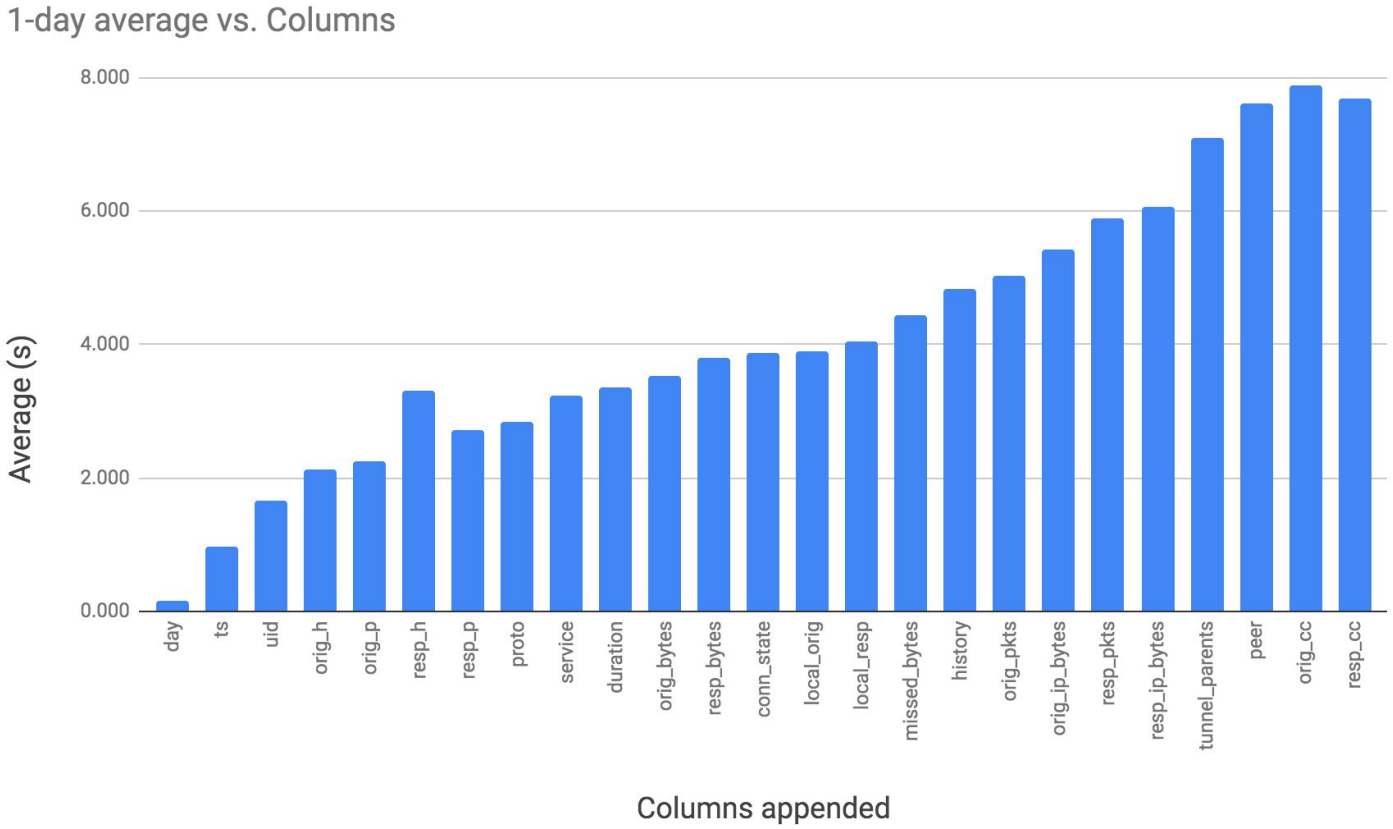
1.1.1.1
Avg: 3.89K / day
Total: 4.06M

8.8.4.4
Avg: 26.33K / day
Total: 28.72M

(all strict queries)

1,095 day data set
728B records

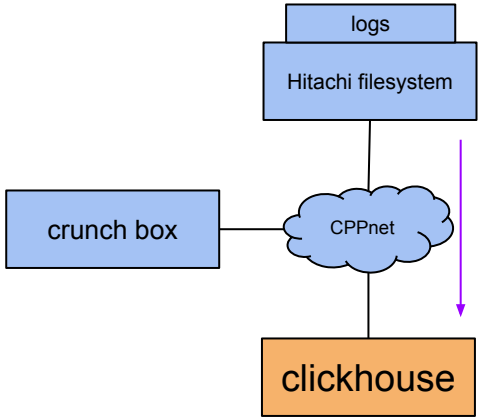
(-C = WHERE ip='x.x.x.x')



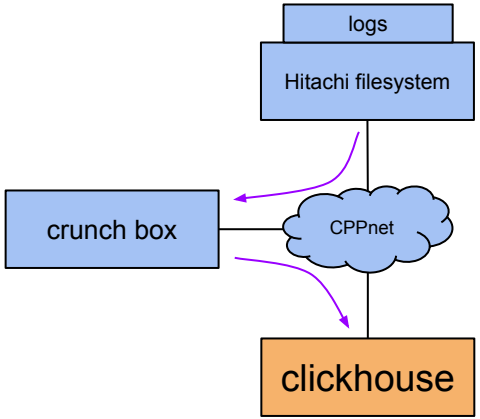
time clickhouse-client --query="select [NxCOLUMNS] from zeek_conn_log_full_nonoptimized order by [Nth COLUMN] desc limit 10"

Architecture

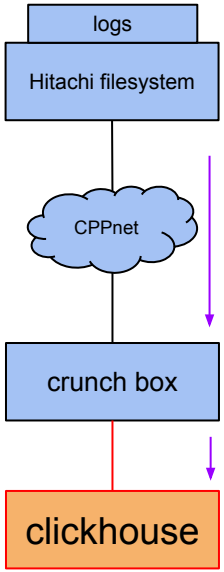
Paranoia Level 0.5



Paranoia Level 1



Paranoia Level 2



locally connected
to crunch box,
no other connectivity

- clickhouse-server and clickhouse-client both run on "clickhouse"
- everything else interacts with it via SSH
- designing for Paranoia Level 2 just in case

Size / Capacity Planning

Flavor of Data	Est. Size	Est. Duration
Date/IP Index only	102 GB	1,095 days (3 yrs)
Date/IP Index only	16 TB	114,213 days (313 yrs)
More log files, full columns	16 TB	800 days (2.2 yrs)
More log files, full columns (search speed optimized by sort key)	16 TB	496 days (1.4 yrs)

Future

- New, faster hardware
- Config tuning (memory, threads)
- More storage, more days of data
- More columns in conn.logs
- Multiple indices?

- Other data sets
  - More Zeek(bro)
    - DNS logs
    - File logs
    - HTTP logs
    - SMTP logs
  - Syslog
    - DNS query logs

Resources

- ClickHouse Docs:
  https://clickhouse.yandex

- Justin Azoff's Clickhouse Talk at BroCon17
  - https://youtu.be/EeW0VXLv6dc?t=684

- NCSA Clickhouse Stuff:
  - https://github.com/ncsa/bro-clickhouse

- Some LBL Scripts and examples:
  - https://github.com/michaelsmitasin/lbl-cpp-clickhouse

Questions? Suggestions?

- DOE NSM Mattermost: lbnl.mnsmitasin
- mnsmitasin@lbl.gov
- security@lbl.gov

Bonus: Google BigQuery?

- (public ~Jul 2018 prices)
- Active Storage: $0.02/GB after 10GB
- Long-Term Storage: $0.01/GB after 10GB
- Streaming Inserts: $0.01/200MB
- Queries: $5/TB after 1TB
- LBL back-of-napkin calcs:
  - 1 year (2017) conn.logs = ~22GB/day, ~8T/year
  - $22K/year for first year of storage (only grows)
  - + $7.5K/year assuming 1 query/day
  - + $420K/year in streaming inserts for "real time"

What else is LBNL working on?

(NextGen Blocking)

What else is LBNL working on?

CLOUDFLARE®

(BOD 18-01 + WAF + Visibility)

What else is LBNL working on?

→

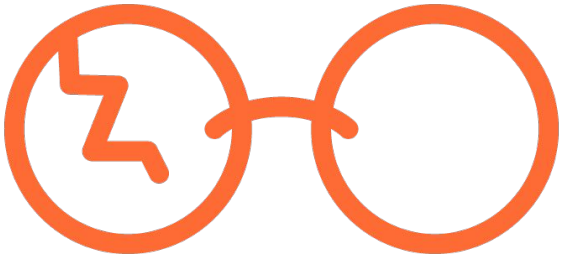
(Firewalls)

What else is LBNL working on?

(Cert Management)

What else is LBNL working on?

The University of Texas at Austin
Information Security Office



(Dorkbot)