Asura: A huge PCAP file analyzer for anomaly packets detection using massive multithreading

DEF CON 26, Aug 12 2018

Ruo Ando

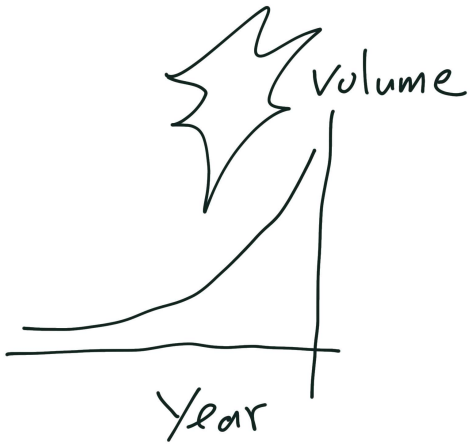
Center for Cybersecurity Research and Development

National Institute of Informatics

Outline

- ”Too many packets, too few resources”
  - 100,000,000 vs 1000,000,000,000
- ASURA: “Huge PCAP file vs Massive threads”
  - Overview
  - Task based decomposition
  - Selection of features and containers
  - Reduction by massive threads
- Demo and Experimental results
- Conclusions

Story behind Asura



Traffic explosion.
Internet Traffic continues to increase at exponential rate, no end in sight.

Too many packets, too few professionals.
Cyber attack has become more sophisticated.

Reference: The Scream @ public domain

100,000,000 vs 1000,000,000,000

- “The universe is not complicated, there's just a lot of it.”

  - Richard Feynman

- Unreasonable Effectiveness of Data

  If a machine learning program cannot work with a training of a million examples, then the intuitive conclusion follows that it cannot work at all.

  However, it has become clear that machine learning using a huge dataset with a trillion items can be highly effective in tasks for which machine learning using a sanitized (clean) dataset with a only million items is NOT useful.

  Chen Sun, Abhinav Shrivastava, Saurabh Singh, Abhinav Gupta,“Revisiting Unreasonable Effectiveness of Data in Deep Learning Era”, ICCV 2017

  https://arxiv.org/abs/1707.02968

Overview of Asura

- **Portable and reasonable**

  GPU, Spark are still high-cost.

- **Posix Pthreads (explicit parallel programming model)**

  Pthreads represent the assembly languages of parallelism.
  Maximum flexibility.

- **Parser – full scratch, without libpcap**

  Flexible. More scalable compared with tshark (in some cases).

- **Compact but powerful**

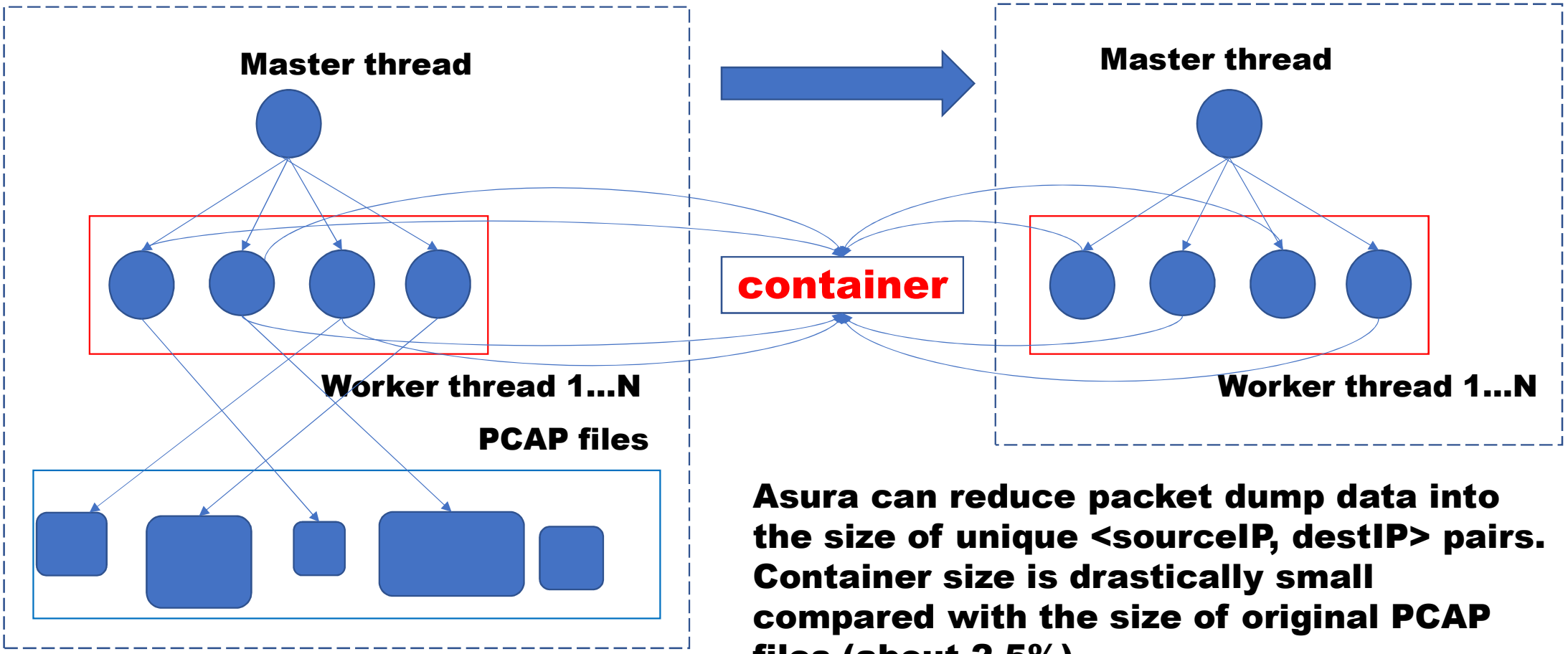
  Asura has thousands lines of code.

  Asura can process 76,835,550 packets in 200-400 minutes.

Asura: Huge PCAP file vs Massive threads

Reduction (task decomposition)

Clustering (data decomposition)



Asura can reduce packet dump data into the size of unique <sourceIP, destIP> pairs. Container size is drastically small compared with the size of original PCAP files (about 2-5%).

Task decomposition vs Data decomposition

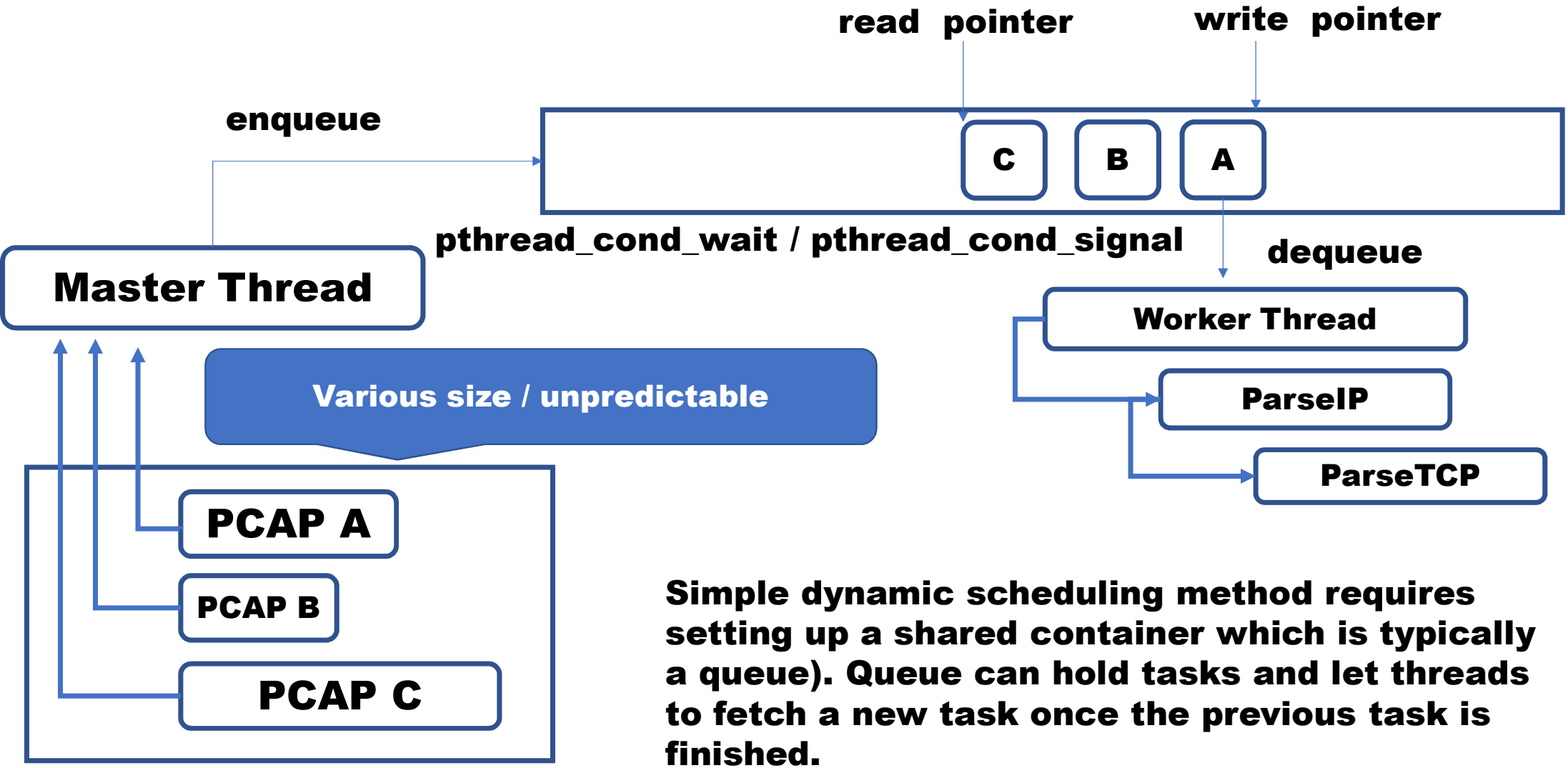
If we want to transform code into a concurrent version, there are two ways.

- **One way is data decomposition, in which the program cope with a large collection of data and can process every chunk of the data independently.**

- **Another way is task decomposition, in which the process is divided into a set of independent task so that threads can run in any order.**

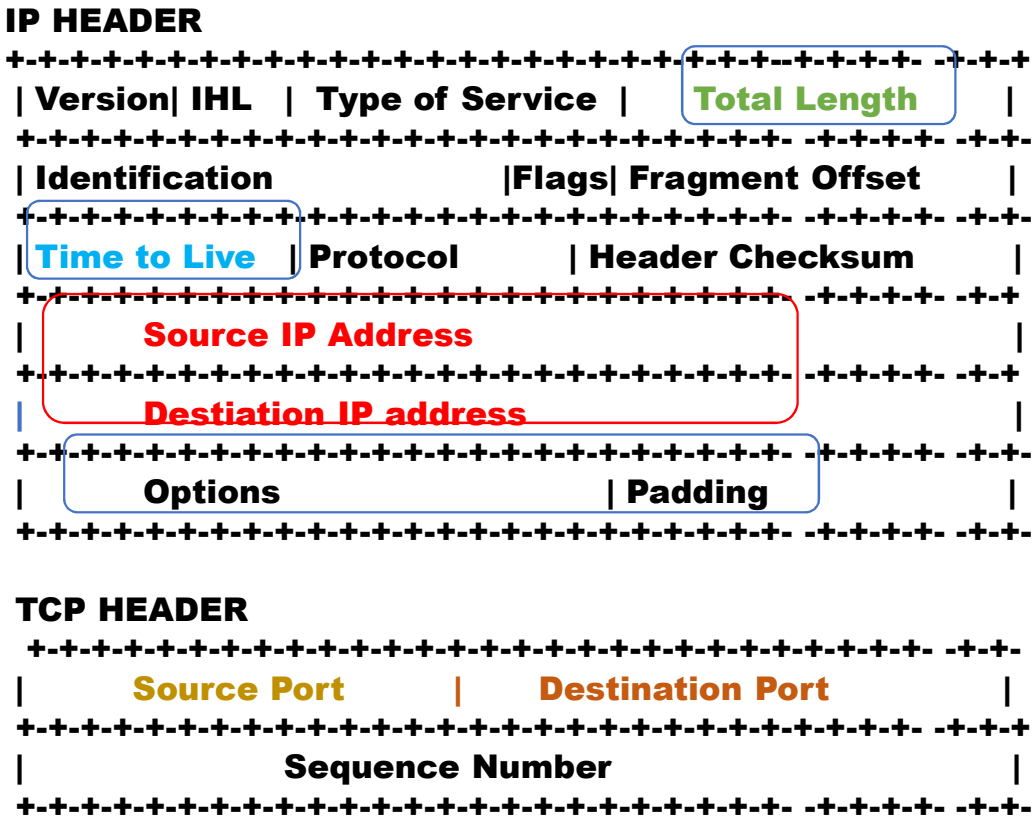
- **As with PCAP file parsing, load balance is an important factor to take into consideration, especially when PCAP files are variable sizes. Real-world PCAP file is NOT organized in a regular pattern and unpredictable.**

Task based decomposition (dynamic scheduler)



Simple dynamic scheduling method requires setting up a shared container which is typically a queue). Queue can hold tasks and let threads to fetch a new task once the previous task is finished.

Feature selection




```
1: /* STRUCTURE I: srcIP, destIP */
2: typedef struct _addrpair {
3: map<string, string> m;
4: pthread_mutex_t mutex;
5: } addrpair_t;
6: addrpair_t addrpair;
```

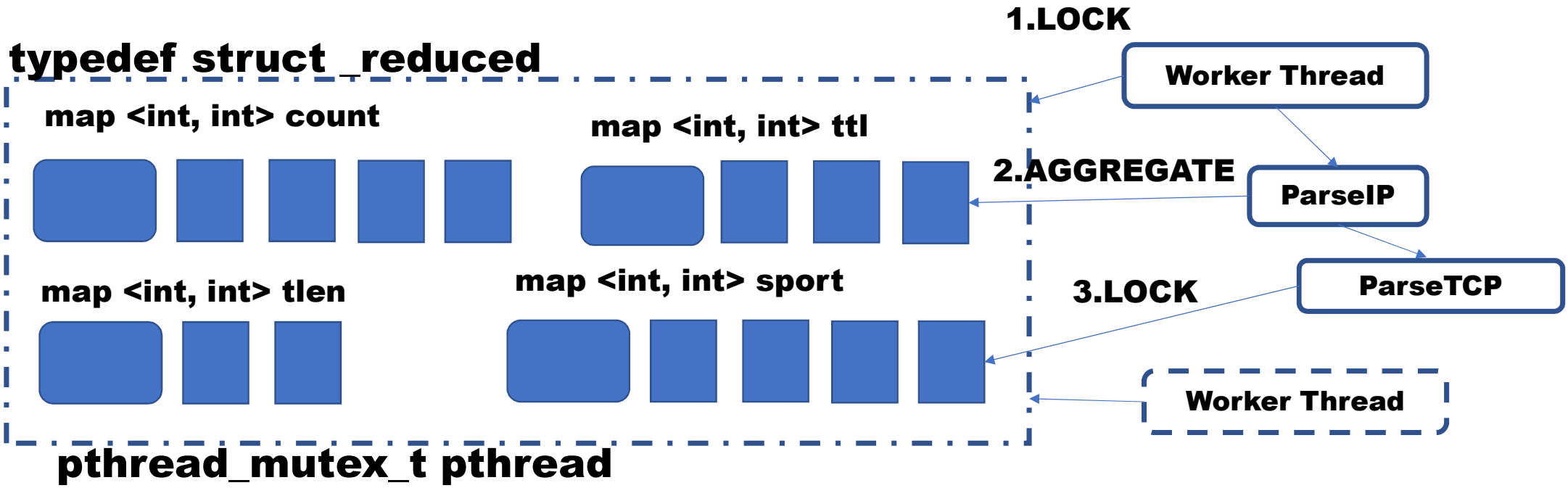
Key **Value**

{<sourceIP, destIP>, V[i]}

```
1: /* STRUCTURE II: reduced */
2: typedef struct _reduced {
3: map<int, int> count;
4: map<int, int> tlen;
5: map<int, int> ttl;
6: map<int, int> sport;
7: map<int, int> dport;
8: pthread_mutex_t mutex;
9: } reduced_t;
10: reduced_t reduced;
```

Asura can reduce packet dump data into the size of unique <sourceIP, destIP> pairs.

Reduction by massive threads



- Procedures of worker thread are intuitively simple:

1. lock struct _reduced
2. aggregate the members of struct _reduced
3. unlock struct _reduced

Choice of container

- **STL + Posix Pthreads: Exposing expose the control of parallelism at its lowest level. Maximum flexibility, but at a high cost in terms of hacker's effort (sometimes painful).**

- **Intel TBB: An excellent library for task-based parallelism mainly for scientific computation. Therefore Input data should be well structured and organized.**

- **Nvidia Thrust: C++ template library for CUDA based on the Standard Template Library (STL). Map < > container has not been implemented yet (as far as I know).**

PCAP files are NOT organized in a regular pattern or the parsing of PCAP files is different or unpredictable for each element in the stream. So STL + Posix Pthreads is the choice.

Experimental results

18G: 76,835,550 packets with max queue size 1024 - about 5 to 6 hours

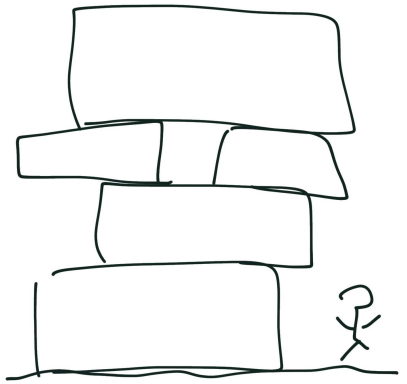
# of threads	time	
1	real	976m46.680s
2	real	474m0.328s
500	real	287m21.413s
1000	real	346m16.257

18G: 76,835,550 packets with max queue size 52 - about 7 to 9 hours

# of threads	time	
200	real	877m9.874s
500	real	464m42.022s
1000	real	493m24.110s
4000	real	523m43.533s

Kernel tuning (Ubuntu 16 LTS)

- **Logical cores N**
  grep processor /proc/cpuinfo | wc –l
- **files and procs - /etc/securitly/limits.conf**
  It should be more than 1024
- **posix queue size - /etc/security/limits.d**



	sourceIP	destIP	rare rate
MALWARE	*.*.*.*	*.*.*.*	0.73%
FINGERPRINTING	*.*.*.*	*.*.*.*	2.36%
BRUTE FORCE SSH	*.*.*.*	*.*.*.*	3.23%
NORMAL	*.*.*.*	*.*.*.*	91.95%

1 minutes Demo

Conclusions

- **Asura is full-scratch (and painful) implementation with POSIX Pthreads and C++ STL.**

- **Leveraging Pthreads which represents assembly language in parallelism provides maximum flexibility.**

- **Asura adopts task-based implementation for processing huge, heterogeneous and unpredictable PCAP file stream.**

- **Asura has thousands lines of code and can process 76,835,550 packets in 200-500 minutes.**

Thank you !

https://github.com/RuoAndo/Asura