DBpia

논문 21-46-06-01 The Journal of Korean Institute of Communications and Information Sciences '21-06 Vol.46 No.06
https://doi.org/10.7840/kics.2021.46.6.929

Elastic-C: 카운터 기반 교체 방식을 이용한 Elastic Sketch

양 승 삼◆, 장 룡 호*, 양 대 헌**, 노 영 태°

Elastic-C: Elastic Sketch Using Counter-Based Swapping

Seung-sam Yang◆, Rhong-ho Jang*, Dae-hun Nyang**, Young-tae Noh°

요 약

네트워크 환경을 원활하게 유지하기 위해서는 데이터 스트림에서 혼잡제어, DoS 공격, 스캔 공격 등의 문제를 유발하는 원인을 파악하는 것이 중요하다. 이를 해결하기 위해 데이터 스트림의 각 플로우의 빈도수를 측정하여 문제를 유발하는 플로우를 찾는 작업이 요구되지만, 제한된 환경에서 데이터 스트림을 정확하고 신속하게 측정하는 것은 어려운 일이다. 최근 발표된 Elastic sketch는 기존 알고리즘과는 다르게 해시테이블을 카운터 앞에 두고 빈도수 투표에 의한 추방 방법(이하 voting)을 사용하여 빈도수가 높은 플로우(elephant flow)를 해시테이블에 따로 저장한다. 그 결과 빈도수가 높은 플로우가 카운터에서 유발할 수 있는 측정오차가 줄어들기 때문에 상당수의 빈도수가 작은 플로우(mouse flow)가 정확해진다. 그러나 이들이 사용한 voting이라는 방법은 해시테이블에서 수행되는 알고리즘이기 때문에 부가적인 메모리와 연산을 요구하는 등 여러 문제가 발생한다. 이 논문에서는 voting 알고리즘이 유발하는 문제점을 분석하고 이를 해결할 수 있는 새로운 방식인 Elastic-C를 소개한다.

Key Words : Sketch, Elastic-C, Counter-based swapping, Traffic measurement, Estimation

ABSTRACT

In order to keep the network environment smooth, it is important to identify the causes of problems such as congestion control, DoS attacks, and scan attacks in the data stream. To solve this problem, it is required to measure the frequency of each flow of the data stream to find the flow causing the problem, but it is difficult to accurately and quickly measure the data stream in a limited environment. Unlike existing algorithms, the recently announced Elastic Sketch uses a hash table in front of a counter and uses the expulsion method by frequency voting (hereinafter, voting) to separately store high-frequency flows in the hash table. As a result, since the measurement error that can be caused by a counter with a high frequency flow is reduced, a significant number of flows with a small frequency become accurate. However, since the voting method they used is an algorithm that is performed on a hash table, several problems arise, such as requiring additional memory and operation. In this paper, we analyze the problems caused by the voting algorithm and introduce Elastic-C, a new method that can solve them.

※ 본 연구는 2019년도 교육부의 재원으로 한국연구재단의 기초연구사업 지원을 받아 수행된 것임 (NRF-2019R1F1A1059898)
◆ First Author : Inha University Computer Science Engineering, seungsam@nsl.inha.ac.kr, 학생회원
° Corresponding Author : Inha University Computer Science Engineering, ytnoh@nsl.inha.ac.kr, 종신회원
* Wayne State University Computer Science, r.jang@wayne.edu, 정회원
** Ewha Womans University Cyber Security, nyang@ewha.ac.kr, 정회원
논문번호 : 202101-018-A-RE, Received January 18, 2021; Revised March 5, 2021; Accepted March 7, 2021

www.dbpia.co.kr

Ⅰ. 서 론

클라우드 컴퓨팅, 엣지 컴퓨팅, IoT, 빅데이터 등 IT산업의 고도화로 네트워크 트래픽은 매년 증가하고 있으며, 이에 따라 방대한 트래픽을 고속으로 처리할 수 있는 장치의 수요는 꾸준히 증가하고 있다[1,2]. 그러나 네트워크에 사용되는 자원이 꾸준히 증가함에도 불구하고 불균형적 트래픽 통신 설계 또는 각종 스캐닝 공격과 DoS 공격 등으로 네트워크 환경에 여러 문제가 발생한다[3-6]. 네트워크의 원활한 환경을 유지하기 위해서는 문제를 유발하는 근원을 찾고 제한하는 방법이 필요하지만, 제한된 메모리로 빠르고 정확히 데이터 스트림을 요약하여 문제를 해결하는 것은 매우 어렵다. 이러한 관점에서 바라볼 때 스케치는 매우 중요한 연구로 볼 수 있다. 최근 T. Yang 등은 해시테이블을 이용하여 확률적으로 elephant flow를 데이터 스트림에서 구별하는 방법인 Elastic sketch[7]를 발표했다. 카운터는 모든 플로우가 공유하는 공간이기 때문에 mouse flow가 elephant flow와 카운터를 공유하게 되면 측정오차가 크게 발생할 수 있는데, 기존의 스케치는 플로우가 입력되면 카운터를 갱신한 후에 최소힙(min heap)이나 해시테이블 같은 자료구조에 elephant flow를 저장하는 방식이었기 때문에 상당수의 mouse flow가 과대측정 될 수 있다. Elastic sketch는 해시테이블을 데이터 스트림에서 elephant flow를 구별하는 필터로 사용하여 카운터에서 발생하는 측정오차를 줄였기 때문에 상당히 정확해진다. 그러나 이들의 방법은 카운터의 정보가 배제되는 알고리즘이기 때문에 부가적인 연산과 메모리를 요구한다. 이 논문에서 Elastic sketch의 문제점을 자세히 다루고, 이를 개선하여 더 정확하고 신속한 스케치를 소개한다. 이 논문에서 수행한 실험에서 기존 Elastic sketch보다 처리율이 약 20% 증가했으며, ARE, RMSE, Heavy Hitter 탐지 등 모든 실험에서 향상된 결과를 얻을 수 있었다.

Ⅱ. 관련연구

스케치는 제한된 메모리로 주어진 데이터에서 필요한 정보를 추출하는 방법이다. 그중 상당수의 알고리즘은 w개의 카운터로 이루어진 d개의 배열을 사용하여 데이터를 기록하는 방법으로 정보를 추출한다. Count sketch[8]는 d개의 각 배열마다 위치를 나타내는 해시와 부호를 나타내는 해시가 있다. 각 배열의 위치 해시로 1개의 카운터가 선택되므로 플로우마다 d개의 카운터를 사용하며 플로우별로 해당하는 카운터의 부호 해시에 따라 카운터 값을 증감한다. 측정값은 d개의 해당 카운터 중 중간값을 선택한다.

Count-Min sketch(CM sketch)[9]는 위치를 나타내는 해시만 존재하며 플로우가 입력되면 각 배열의 선택된 카운터를 증가시킨다. CM sketch는 매 입력마다 카운터의 값을 증가하기 때문에 모든 카운터가 플로우의 빈도수보다 크거나 같다. 따라서 해당 카운터 중 최솟값을 측정값으로 선택한다.

그림 1은 Conservative update(CU sketch)[10]의 예제로 CM sketch에서 선택된 카운터들 중에 최솟값을 갱신하여 얻은 값 m보다 작은 카운터만 m으로 갱신하는 방법이다. 예제에 나타낸 것은 갱신할 값이 1인 경우이며, 카운터들 중 최솟값은 3이므로 4보다 작은 카운터만 4로 갱신된다. 이 방법으로 갱신을 하면 모든 카운터를 동일하게 증가하지 않더라도 플로우의 빈도수보다 작은 경우는 발생하지 않으면서도 측정오차를 줄일 수 있다.

그러나 이러한 초기 스케치 연구들은 Heavy Hitter 탐지와 같이 플로우의 정보 등 여러 부가 정보가 필요한 문제를 고려하기에는 적합하지 않아 최소힙이나 해시테이블과 같은 자료구조가 추가로 필요했으며 이들은 매우 단순히 elephant flow를 기록하는 목적으로 사용되었다.

반면, Elastic sketch[7]는 해시테이블을 단순히 기록 목적으로 사용하지 않고 카운터 앞에 두어 elephant 필터로써 사용하였다. 이들이 사용한 voting 알고리즘은 해시테이블에서 elephant flow를 선별하는 알고리즘으로 특정 조건이 발생하면 플로우 정보를 교체한다. 해시테이블에 기록되지 않은 플로우의 정보만 카운터에 기록되기 때문에 카운터에 기록되는 elephant flow의 수가 줄어들어 발생하는 측정오차를 상당히 줄일 수 있다.

그러나 voting 방법은 해시테이블에서 동작하기 때

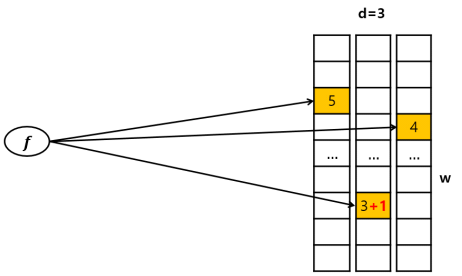

그림 1. Conservative Update 예제
Fig. 1. Conservative Update example.

www.dbpia.co.kr

문에 카운터의 정보는 배제된다. 이로 인해, 부가적인 메모리와 연산이 필요하게 되어 정확도와 속도가 떨어진다. 다음 장에서 기존 Elastic sketch 알고리즘과 그에 따른 문제점을 알아보고 이에 대한 해결책으로 새로운 방법인 Elastic-C를 제시한다.

Ⅲ. 카운터 기반 교체 방식을 이용한 Elastic

3.1 기존의 Elastic sketch

구조. 그림 2는 기존의 Elastic sketch[7]의 소프트웨어 버전을 나타낸 그림으로 해시테이블은 S개의 슬롯으로 이루어진 B개의 버킷을 의미하며, 각 슬롯은 플로우의 ID인 key와 해당 플로우의 빈도수인 vote+, 플로우 key정보의 교체 여부를 나타내는 flag로 이루어져 있고, 버킷의 맨 마지막 슬롯에는 해시테이블에 기록되지 않은 다른 flow들의 빈도수 총합을 나타내는 vote-가 있다. vote+와 flag는 하나의 메모리에 있으며 flag는 이 메모리의 MSB로 나타낸다. flag의 초깃값은 False(MSB=0)이고 슬롯에서 교체가 발생하면 True(MSB=1)가 된다. 주어진 예제에서 S=3, B=4이다.

입력. 플로우가 입력되면 Elastic sketch는 해시 값에 해당하는 버킷의 모든 슬롯을 순차적으로 탐색하며 아래의 조건에 따라 갱신을 한다.

1) 빈 슬롯이 있는 경우, 처음 발견한 빈 슬롯의 key를 입력된 플로우의 key로 기록하고, vote+를 1로 설정한다.

2) 입력된 플로우가 해시테이블에 있는 경우, 해당 슬롯의 vote+를 1 증가한다.

3) 입력된 플로우가 해시테이블에 없는 경우, vote-를 증가한다. 만약 해당 버킷의 슬롯 중 최소 vote+값을 갖는 플로우가 미리 정해두었던 경계값(λ)과 곱하여 vote- 보다 크다면 해시테이블의 변화는 없고 입력된 플로우에 해당하는 카운터를 증가한다.

4) 입력된 플로우가 해시테이블에 없는 경우, vote-를 증가한다. 만약 해당 버킷의 슬롯 중 최소 vote+값을 갖는 플로우가 미리 정해두었던 경계값(λ)과 곱해도 vote- 보다 작거나 같다면 해당 슬롯의 key 정보를 입력된 플로우의 key로, vote+는 1, flag는 True로 변경하고 vote-는 0으로 초기화 한다. 이후 슬롯에서 제거된 플로우는 제거되기 직전까지의 vote+값만큼 해당 카운터의 값을 증가한다.

출력. 먼저 해시테이블에서 해당 버킷을 탐색하여 기록되어 있는지 확인하고, 해시테이블에 기록되어 있다면 flag를 확인한다. 만약, flag가 False라면 교체된 적이 없으므로 vote+를 그대로 출력하며, True라면 카운터에서 측정되는 값을 vote+와 더하여 출력한다. 만약, 해시테이블에 기록이 되지 않았다면 카운터에서 측정되는 값을 그대로 출력한다.

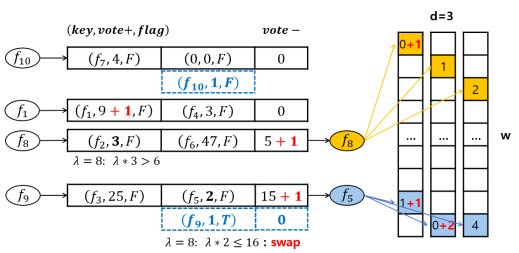

그림 2. 기존의 소프트웨어 버전의 Elastic sketch
Fig. 2. Elastic sketch of original software version

Elastic의 문제점. 앞서 언급한 바와 같이 voting기법은 해시테이블에서 동작하기 때문에 카운터의 정보는 배제되어 다음과 같은 문제점이 발생한다.

1) 불필요한 메모리 할당: voting을 하기 위해서 vote-와 같은 메모리를 따로 할당해야 한다.

2) 불필요한 연산 추가: 해시테이블에서 교체가 발생한 슬롯에서는 기록된 추정치가 실제 빈도수보다 작을 수 있어 플로우 정보의 교체 여부를 표기하기 위해 flag를 사용해야 한다. 또한 해시테이블에서 빈도수를 구하기 위해서는 vote+에서 flag를 제거하는 연산이 요구된다.

3) 오차의 누적: flag가 True인 슬롯에 기록된 플로우는 카운터의 값을 추가하기 때문에 해시테이블에 기록된 후에도 다른 플로우 때문에 카운터에서 발생하는 추가적인 측정오차가 누적된다.

3.2 카운터 기반 교체 방식 Elastic sketch

구조. 그림 3은 Elastic-C의 구조로 각 슬롯이 플로우의 ID인 key와 해당 플로우의 빈도수인 val로 이루어져 있으며 vote- 없이 모든 슬롯이 플로우의 정보를 기록하는 슬롯으로 사용된다.

입력. 탐색 방법은 기존과 같으며 아래의 조건에 따라 갱신을 한다.

1) 빈 슬롯이 있는 경우, 처음 발견한 빈 슬롯의 key를 입력된 플로우의 key로 기록하고, val를 1로 설정한다.

2) 입력된 플로우가 해시테이블에 있는 경우, 해당 슬롯의 val를 1 증가한다.

3) 입력된 플로우가 해시테이블에 없는 경우, 카운터에서 얻은 측정값이 해당 버킷의 최소 val보다 작으

www.dbpia.co.kr

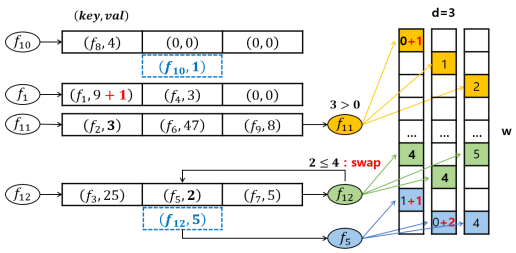

그림 3. 카운터 기반 교체 방식을 이용한 Elastic sketch
Fig. 3. Elastic sketch using counter-based swapping

면 입력된 플로우의 카운터를 갱신한다.

4) 입력된 플로우가 해시테이블에 없는 경우, 카운터에서 얻은 측정값이 해당 버킷의 최소 val보다 크거나 같은 경우, 해당 슬롯의 key 정보를 입력된 플로우의 key로, val는 카운터의 측정값+1로 변경한다. 이후 슬롯에서 제거된 플로우는 해당 카운터 중에 제거되기 직전의 val보다 작은 카운터만 val로 갱신한다.

출력. 먼저 해시테이블에서 해당 버킷을 탐색하여 기록되어 있는지 확인하고, 해시테이블에 기록되어 있다면 val를 출력하며, 기록되지 않았다면 카운터에서 측정되는 값을 그대로 출력한다.

Elastic-C의 특징. Elastic-C는 기존의 Elastic과 비교하여 다음과 같은 특징 때문에 더 정확하고 신속하게 측정할 수 있다.

1) 누적오차 감소: Elastic-C는 vote-와 같은 공간이 없어 더 많은 elephant flow의 정보를 해시테이블에 저장할 수 있다. 따라서 더 많은 flow 정보가 카운터에 기록되지 않아도 되기 때문에 카운터에서 발생하는 오차가 줄어든다. 또한, 기존 Elastic에서는 해시테이블의 임의의 슬롯에서 플로우 정보가 교체된 적이 있는 경우, 해시테이블에서 얻은 추정치와 카운터에서 얻은 추정치를 누적하여 결과를 도출했지만, Elastic-C의 해시테이블의 모든 슬롯의 추정치는 빈도수보다 항상 크거나 같으므로 카운터에서 얻은 추정치를 누적할 필요가 없다. 즉, 해시테이블에 기록된 이후에는 카운터에서 발생하는 오차가 최종 측정에 더해지지 않는다.

2) 플로우의 중첩도 감소: 슬롯의 정보를 교체할지 결정할 때 Elastic은 vote-의 정보를 사용하지만, Elastic-C는 카운터의 정보를 사용한다. vote-는 해시테이블에 할당된 공간이므로 단위 공간당 요구되는 메모리의 양이 많고, 일반적으로 해시테이블보다 카운터에 많은 메모리를 할당하지만, 해시테이블에 더 많은 메모리가 할당되더라도 vote-의 공간은 해시테이블의 일부이기 때문에 카운터의 개수가 vote-의 개수보다 훨씬 많을 수 있다. 예를 들어 버킷당 슬롯의 개수가 8개인 해시테이블이고, 204KB(약 0.2MB) 중에 해시테이블로 150KB를 사용하더라도 카운터의 개수와 vote-의 개수는 다음과 같다. (key와 val는 모두 4Byte인 경우로 계산하였다.)

카운터 개수 : 54KB / 4Byte = 13824

vote-의 개수 : 150KB / 8 / 8Byte = 2400

따라서 Elastic-C를 사용하면 해시테이블의 정보를 교체할 때 사용하는 공간이 많으므로 같은 공간을 공유하는 플로우의 수가 감소하고 이것은 더 정확한 측정이 가능함을 의미한다.

3) 연산량 감소: Elastic-C는 flag를 사용하지 않기 때문에 부가 연산이 없이 해시테이블에 있는 플로우의 빈도수를 구할 수 있어 기존의 Elastic보다 빠르게 해시테이블을 탐색할 수 있다.

Ⅳ. 평 가

이 장에서는 Elastic-C를 CM sketch, CU sketch와 기존의 Elastic sketch와 비교한다.

4.1 실험환경

데이터 및 실험환경. CAIDA의 Equinix Chicago 데이터 센터에서 수집된 1시간 데이터를 사용하였다[11]. 이 데이터를 1분 단위로 나누었으며 각 데이터에는 약 32M개의 패킷이 있고, 약 350K개의 서로 다른 플로우가 존재한다. 각 플로우의 key는 srcIP로 하였다.

스케치 종류와 평가 항목.

- 플로우별 빈도수 추정 그래프: x축은 플로우의 빈도수 f_i이고 y축은 플로우의 측정값 $\hat{f}_i$이며 플로우의 정보를 평면 위의 한 점 $(f_i, \hat{f}_i)$로 나타낸 그래프이다. 플로우가 y=x축과 가까울수록 정확하게 추정된 것이다.
- ARE(Average Relative Error): 서로 다른 n개의 플로우의 빈도수(f_i)와 추정치($\hat{f}_i$)의 상대오차의 평균값으로 $\frac{1}{n}\sum_{i=1}^{n}\frac{|\hat{f}_i - f_i|}{f_i}$이다.
- RMSE(Root Mean Square Error): 서로 다른 n개의 플로우의 빈도수(f_i)와 추정치($\hat{f}_i$)의 편차 제곱 평균의 제곱근으로 $\sqrt{\frac{1}{n}\sum_{i=1}^{n}(\hat{f}_i - f_i)^2}$이다.
- Heavy Hitter 탐지율: 전체 packet의 0.01% 이상

www.dbpia.co.kr

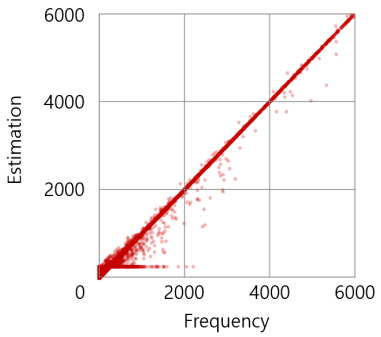

그림 4. 8bit 카운터를 사용하는 Elastic sketch의 플로우 빈도수 추정 그래프
Fig. 4. Flow frequency estimation graph in Elastic sketch using 8-bit counter.

입력된 플로우를 heavy hitter로 정하여 F1 score, Precision, Recall을 구했다. CM sketch, CU sketch의 경우 큰 플로우를 저장하는 공간을 각각 두 종류를 최소힙(min heap)과 해시테이블 두 종류로 나누어 실험했다.

- 처리율: 데이터 스트림을 처리하는데 걸리는 시간으로 단위는 (Mpps)이다. CPU에서 해시테이블은 병렬처리기법인 SIMD(Single Instruction Multiple Data)를 적용하여 가속할 수 있다.

스케치 설정. 모든 스케치의 카운터 배열의 개수는 3개이며 해시테이블이나 최소힙의 메모리는 150KB를 사용하였다. Elastic sketch 논문에서 제안한 알고리즘은 8bit 카운터로 이루어진 한 줄의 카운터를 사용했지만 데이터가 많아지면 8bit로는 충분한 정보를 기록할 수 없기 때문에 그림 4와 같이 과소측정이 되는 현상이 발생한다. 이는 네트워크 환경에서 공격자를 일반 사용자로 잘못 판단하는 심각한 문제를 유발할 수도 있다. 이 논문에서는 앞서 언급한 모든 알고리즘에 32bit로 이루어진 카운터를 동일하게 사용하여 과소측정이 되는 경우는 발생하지 않도록 하였다. 기존 Elastic sketch와 Elastic-C의 카운터 부분은 CU sketch[10]를 사용하였다.

4.2 정확도

플로우별 빈도수 추정 그래프. 그림 5는 0.2MB를 사용하여 얻은 플로우별 빈도수 추정 그래프이다. 각 그래프는 CM, CU, Elastic, Elastic-C 순이다. CM과 CU는 모든 플로우가 카운터를 공유하기 때문에 mouse flow임에도 상당히 크게 측정되는 플로우가 다수 발생했지만, Elastic sketch는 다수의 elephant flow가 해시테이블에서 걸러지기 때문에 측정오차가 과도하게 발생하는 경우가 일어나지 않은 것을 확인할 수 있다. 더욱이 Elastic-C는 앞서 언급한 것처럼 해시테이블에 기록된 이후로는 카운터에서 발생하는 측정오차와는 무관하기에 기존의 Elastic보다 측정오차가 더 작게 나타난다.

ARE(Average Relative Error). 그림 6은 메모리별로 구한 ARE로 0.2MB를 제외하고는 Elastic-C가 가장 정확한 것으로 나타났다. 0.2MB일 때는 Elastic의 카운터가 54KB이기 때문에 상당수의 elephant flow가 해시테이블에서 구별되더라도 204KB를 사용한 CM, CU보다 다소 크게 측정된 것으로 보인다. 기존의 Elastic과 비교해보면 Elastic-C는 ARE 측면에서 2.2% ~ 27.3%의 개선효과를 보였다.

RMSE(Root Mean Square Error). 그림 7은 메모리별로 구한 RMSE로 실험한 모든 경우에 Elastic-C가 정확한 것으로 나타났다. RMSE는 편차의 제곱을 다루기 때문에 빈도수와 측정값의 차이가 큰 플로우가 클수록 값이 크게 나타난다. 따라서 Elastic-C는 전반적으로 플로우의 편차가 작다는 것을 의미한다. 기존의 Elastic과 비교해보면 Elastic-C는 RMSE 측면에서 6.5% ~ 28.6%의 개선효과를 보였다.

Heavy Hitter 탐지율. 그림 8은 메모리별로 구한 Heavy Hitter 탐지율이며, 위의 그래프에서 T는 해시테이블, H는 최소힙을 의미하며 CM sketch와 CU sketch에서 elephant flow를 저장하기 위해 사용한 자료구조를 의미한다. 실험한 모든 경우에 Elastic-C가

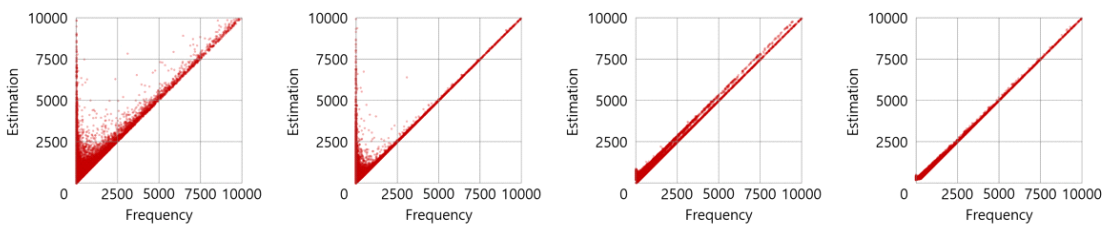

그림 5. 0.2MB 메모리로 측정한 플로우별 빈도수 추정 그래프(CM, CU, Elastic, Elastic-C)
Fig. 5. Frequency estimation graph per flow measured with 0.2MB memory(CM, CU, Elastic, Elastic-C)

www.dbpia.co.kr

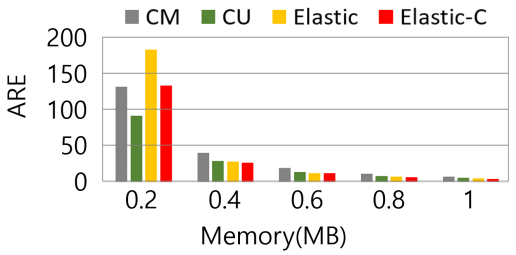

그림 6. ARE 정확도 비교
Fig. 6. Accuracy comparison of ARE

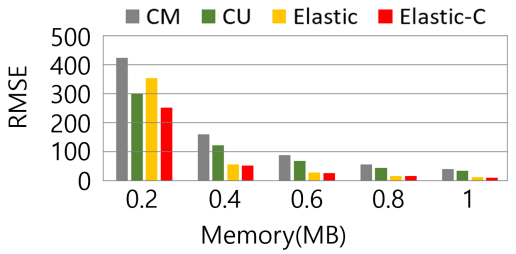

그림 7. RMSE 정확도 비교
Fig. 7. Accuracy comparison of RMSE

정확한 것으로 나타났다. 특히, 99%의 Precision을 달성하기 위해 기존의 Elastic은 0.4MB정도의 메모리가 필요했지만, Elastic-C는 0.2MB만으로 Precision이 99%를 초과하였다.

이러한 결과는 앞서 언급한 Elastic-C의 특징 때문에 나타난다. 빈도수와 추정치 간 오차가 감소하기 때문에 ARE와 RMSE는 감소하고, 오차와 중첩되는 플로우의 개수가 감소하면 해시테이블에서 과도하게 측정되거나 잘못된 교체가 발생할 확률이 줄어들기 때문에 Heavy Hitter 탐지의 정확도는 증가한다.

4.3 처리율

그림 9는 각 실험에서 사용한 모든 sketch의 처리율을 나타낸 것으로 사용된 메모리는 600KB이다. 주어진 sketch 중 Elastic-C의 처리율이 가장 높았으며, 기존 Elastic보다도 약 20% 더 빠른 것으로 나타났다. 이것은 flag를 사용하지 않아 전체적인 연산량이 감소

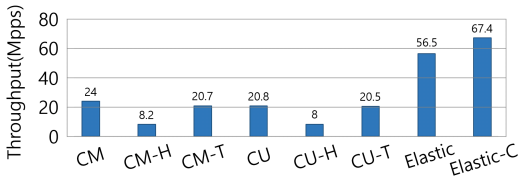

그림 9. 실험에서 사용한 모든 스케치의 처리율
Fig. 9. Throughput of all sketches used in the experiment

했기 때문이다.

V. 결 론

Elastic sketch는 기존 알고리즘과는 다르게 해시테이블을 필터로 사용하여 효과적으로 데이터 스트림을 측정할 수 있었다. 그러나 voting 방법이 카운터의 정보를 배제하기 때문에 앞서 언급한 바와 같이 여러 문제를 유발하였다. 이 논문에서는 카운터의 정보를 포함하여 해시테이블을 관리하는 Elastic-C를 소개하였고, 기존 방식의 문제점을 보완하여 더 빠르고 정확한 측정이 가능함을 보였다.

References

[1] *Cisco Annual Internet Report (2018–2023) White Paper*, Retrieved Dec. 30, 2020, from https://www.cisco.com/c/en/us/solutions/executive-perspectives/annual-internet-report/index.html

[2] J. T. Park, S. M. Cheon, and S. J. Go, "사물인터넷 기반 헬스케어 서비스 및 플랫폼 동향," *KICS Inf. and Commun. Mag.*, vol. 31, issue 12, pp. 25-30, Nov. 2014.

[3] J. Zhang, F. Richard Yu, S. Wang, T. Huang, Z. Liu, and Y. Liu, "Load balancing in data center networks: A survey," *IEEE Commun. Surv. & Tuts.*, vol. 20, issue. 3, pp. 2324-

그림 8. Heavy Hitter 탐지의 정확도 비교
Fig. 8. Accuracy comparison of Heavy Hitter detection

www.dbpia.co.kr

2352, Mar. 2018.

[4] T. Benson, A. Anand, A. Akella, and M. Zhang, "MicroTE: fine grained traffic engineering for data centers," in *Proc. ACM CoNEXT*, pp. 1-12, Tokyo, Japan, Dec. 2011.

[5] G. Shang, P. Zhe, X. Bin, H. Aiqun, and R. Kui, "FloodDefender: Protecting data and control plane resources under sdn-aimed dos attacks," in *Proc. IEEE INFOCOM*, pp. 1-9, Atlanta, GA, USA, May 2017.

[6] S. K. Patel and A. Sonker. "Rule-based network intrusion detection system for port scanning with efficient port scan detection rules using snort," *Int. J. Future Generation Commun. and Netw.*, vol. 9, no. 6, pp. 339-350, Jun. 2016.

[7] T. Yang, J. Jiang, P. Liu, Q. Huang, J. Gong, Y. Zhou, R. Miao, X. Li, and S. Uhlig, "Elastic sketch: Adaptive and fast network-wide measurements." in *Proc. ACM SIGCOMM*, pp. 561-575, Budapest, Hungary, Aug. 2018.

[8] M. Charikar, K. Chen, and M. Farach-Colton, "Finding frequent items in data streams," *ICALP*, pp. 693-703, Malaga, Spain, Jul. 2002.

[9] G. Cormode and S. Muthukrishnan, "An improved data stream summary: the count-min sketch and its applications," *J. Algorithms*, vol. 55, no. 1, pp. 58-75, Apr. 2005.

[10] C. Estan and G. Varghese, "New directions in traffic measurement and accounting: Focusing on the elephants, ignoring the mice," *ACM TOCS*, vol. 21, no. 3, pp. 270-313, Aug. 2003.

[11] *The cooperative association for internet data analysis*, Equinix Chicago data center, Retrieved Dec. 2020, from https://www.caida.org.

양 승 삼 (Seung-sam Yang)

2019년 2월 : 인하대학교 수학과 학사, 컴퓨터 공학과 복수전공
2019년 3월~현재 : 인하대학교 전기컴퓨터공학 석사과정
<관심분야> 네트워크 시스템 및 보안, 암호이론

[ORCID:0000-0002-0938-1570]

장 룡 호 (Rhong-ho Jang)

2013년 8월 : 인하대학교 컴퓨터 정보공학과 학사
2015년 8월 : 인하대학교 컴퓨터 공학과 석사
2020년 2월 : 인하대학교 컴퓨터 공학과 박사
2020년 8월 : University of Central Florida 컴퓨터 공학과 박사
2020년 8월~현재 : Wayne State University 컴퓨터 과학과 조교수
<관심분야> 네트워크 시스템 및 보안, 프라이버시, 악성코드 분석, AI 응용 어플리케이션

[ORCID:0000-0002-3417-6851]

양 대 헌 (Dae-hun Nyang)

1994년 2월 : 한국과학기술원 과학기술 대학 전기 및 전자공학과 학사
1996년 2월 : 연세대학교 컴퓨터과학과 석사
2000년 8월 : 연세대학교 컴퓨터과학과 박사
2000년 9월~2003년 2월 : 한국전자통신연구원 정보보호연구본부 선임연구원
2003년 2월~2020년 2월 : 인하대학교 컴퓨터 공학과 교수
2020년 2월~현재 : 이화여자대학교 사이버보안학과 교수
<관심분야> AI 보안, AI 공격, 네트워크 보안, 개인 정보 보호, 암호 프로토콜

[ORCID:0000-0001-5183-891X]

www.dbpia.co.kr

노 영 태 (Young-tae Noh)

2004년 8월 : 조선대학교 전자계산학과 학사

2007년 2월 : 광주과학기술원 정보기술공학부 석사

2007년 9월 : 광주과학기술정보기술원 정보통신공학과 연구원

2012년 6월 : University of California 컴퓨터 과학과 박사

2012년 7월~2014년 11월 : Cisco Systems Software Engineer

2015년 2월~2015년 8월 : Purdue University Post doc

2015년 9월~현재 : 인하대학교 컴퓨터 공학과 교수

<관심분야> 데이터 센터 네트워킹, 무선 네트워킹, HCI(Human-computer interaction), 머신러닝

[ORCID:0000-0002-9173-1575]

www.dbpia.co.kr