패킷 여과 시스템의 성능 향상을 위한 데이터 캐쉬 잠금 방안 연구

조학봉*, 최창석*, 문종욱*, 정기현*, 최경희**
*아주대학교 전자공학부
**아주대학교 정보통신 전문 대학원
e-mail : sense@csl.ajou.ac.kr

A Study on data cache locking policy for Packet Filtering System's performance improvement

Hak-bong Cho*, Chang-seok Choi*, Jong-wook Moon*,
Gihyun Jung*, Kyunghee Choi**
*Dept. of Electronics Engineering, Ajou University
**A Professional Graduate School
for Information and Communication Engineering, Ajou University

요 약

오늘날 네트워크 보호를 위해 firewall 과 같은 패킷 여과 시스템이 많이 보급되어 있다. 이러한 시스템에서는 해당 패킷의 생사 및 진행방향을 정할 수 있는 Rule 이 다수 존재하며, 각 패킷에 해당하는 Rule 을 검색하는 시간은 전체 네트워크의 응답시간을 지연시킨다. 더불어 해당 네트워크의 병목현상을 일으키는 주범이 될 수 있다. 본 논문에서는 데이터 캐쉬 잠금 방법을 활용한 네트워크 프로세서를 모델로, 캐쉬 잠금을 이용해 패킷 여과 Rule 의 접근 시간을 줄일 수 있는 파라미터를 찾고 수식화하며 Simulation 을 통해 효용성을 검토해 본다.

1. 서론

최근 갈수록 증가하는 현재의 네트워크 트래픽 상황 하 에서 패킷 여과 시스템은 점점 더 고성능을 요하고 있다. 이런 패킷 여과 시스템에서 Filtering Rule 의 조회에 드는 소요시간은 시스템의 성능에 커다란 영향을 끼치는 요소이다. 100M bandwidth 의 네트워크 환경에서는 64byte 의 패킷은 이론적으로 초당 150,000 개의 패킷 이 패킷 여과 시스템으로 들어 올 수 있다. 이런 상황에서 각 패킷마다 적용되는 룰을 찾는데 소요되는 Table lookup cost 는 시스템의 성능에 지대한 영향을 미치게 된다. 이런 Filter Rule table 의 lookup cost 를 줄이기 위해 여러 방법들이 연구 되고 있다. 이중 한 방법으로 cache 를 사용하는 방법이 있다.

캐쉬는 거의 모든 마이크로프로세서에서 성능 향상을 위해 채용하고 있는 메모리 구조이다. 빠른 마이크로프로세서의 속도에 맞춰 외부 주 메모리 속도도 향상되고 있지만, 그 속도 차이는 여전히 무시할 수 없을 정도로 벌어져 있다. 이에 캐쉬는 마이크로프로세서 와 외부 메모리 간의 access 속도를 보정할 수 있도록 고속으로 설계되어 있다. 또한 최근 몇몇의 마이크로프로세서는 더 향상된 캐쉬 사용을 위한 캐쉬 잠금 방법을 지원하고 있다.

캐쉬 잠금 방법은 자주 쓰이는 항목을 미리 캐쉬에 올려두고 locking 을 하여 locking 된 항목은 더 이상 캐쉬에서 replacement 되지 않게 하는 것을 말한다. 즉 캐쉬에 실리게 되고 locking 된 항목의 접근은 항상 cache hit 할 수 있도록 하는 것을 말한다. 반대로 캐쉬에 올라가 있지 않은 항목의 접근은 cache miss 가 발생할 확률이 높아지게 된다. 따라서 시스템의 성격과 환경에 따라 캐쉬 잠금 방법은 시스템의 성능에 득이 될 수 도 있고 실이 될 수 도 있다. 캐쉬 잠금을 지원하는 프로세서는 대부분 캐쉬 전 영역의 locking 과 way 단위의 locking 을 지원한다.

이러한 캐쉬의 사용에 있어서 패킷 여과 시스템에서는 네트워크 에 흐르는 패킷의 종류와 빈도에 따라 적절한 캐쉬 관리 정책을 사용하여야 한다. locality 는 두 가지 관점에서 정의 할 수 있다. 하나는 temporal locality 로서 현재 사용된 것이 가까운 미래에 다시 사용된다는 시간적 의미의 locality 이고 다른 하나는 spatial locality 로서 공간적으로 근처에 위치한 것을 다시 사용할 확률이 높다는 것이다. 네트워크 에 돌아다니는 패킷에서도 이런 locality 를 볼 수 있다[1][2].

이런 locality 는 종종 캐쉬의 활용 방법과 함께 생각되어진다[3][4]. 이 선행 연구에서는 Gateway, Router 에서 IP address 의 locality 를 이용한 캐쉬의 design 을 다루고 있다. 이 논문의 결과는 IP address 의 spatial locality 특성을 이용하여 캐쉬를 design 할 때 replacement 정책은 LRU 에서 최적의 성능을 보였고 캐시의 크기가 커질수록 cache hit 율이 많아지는 것을 보여주고 있다. 네트워크 패킷의 locality 에 관련된 보고서[5]를 보면 패킷 여과 시스템이 설치될 특정 위치에 흐르는 패킷은 locality 를 가지고 있음을 보이고 있다. 예를 들면 웹서버가 설치된 곳에서 패킷의 흐름을 보면 특정 웹 문서를 접근 하는 패킷이 Zipf's law 를 따른다는 연구가 있다[5]. Zipf's law 의 원래 의미는 다음과 같다. 문헌에서 발견되는 용어들의 분포에 대한 다음과 같은 규칙을 말한다. 빈도수* 순위 = 상수이다. 빈도수는 주어진 문헌내의 용어의 출현 빈도를 나타내며, 순위는 다른 용어에 대한 특정 용어의 비중을 나타낸다. 보통 가장 빈도수가 높은 단어의 순위를 1 로 정한다. 즉 순위가 높은 항목의 빈도수가 가장 높다는 법칙이다. 이를 패킷 여과 시스템에서 패킷의 locality 에 적용하면 특정한 몇 가지 Rule 이 가장 높은 빈도수를 가지고 접근 된다는 것을 생각할 수 있다.

본 논문에서는 패킷의 locality 를 고려할 때 캐쉬의 locking way 의 수와 locking 주기 그리고 Filter Rule 의 수를 변화시켜서 각 상황에서의 임계 치를 구하고 이어서 최적의 성능을 낼 수 있는 캐쉬 잠금 정책을 제안하고자 한다.

본 논문의 구성은 서론에 이어서, 2 장에서 패킷의 locality 에 대해 본 논문에서 사용한 Zipf's law 와 캐쉬 관리에 따른 시스템의 성능을 수식을 세워서 Simulation 을 해본다. 3 장에서는 Simulation 의 결과를 분석하고 이를 바탕으로 최적의 캐쉬 관리 정책을 찾아본다. 끝으로 4 장에서는 본 논문에서 도출된 결론과 향후 해결과제에 관하여 논한다.

2. Simulation

2-1 개요

Simulation 환경은 1way 의 크기가 512byte 인 8-way 캐쉬를 모델로 하며, replacement 정책은 LRU 를 따른다. 또한 패킷 여과 시스템을 가정해 한 Rule 의 크기를 32byte 로 두며 이는 cache line size 와 동일하다고 가정한다.

2-2 parameter 설명

첫째 파라미터는 Total Rule 수로서 이 파라미터는 Zipf's law 를 따른 Rule 의 배치와 고정된 크기의 캐쉬 내의 Rule access frequency 는 관련이 있다.

둘째 locking way 수는 고정된 크기의 cache way locking 과 그 시스템의 성능은 관련이 있으며, 셋째 locking 주기 locking overhead 로 인한 시스템 저하와 관련이 있다.

2-3 수식 모델링

수식을 모델링 하는데 있어 먼저 두 가지의 가정을 둔다. 첫째, 패킷 스트림은 Zipf's law 를 따른다. 둘째, 모든 Rule 의 크기는 32bytes 로 모두 동일 하다.

먼저 첫째 가정인 Zipf's law 의 수식은 다음과 같다.

$$f(i) = c/i, (\text{단 } c = 1/\sum_{i=1}^{R}\frac{1}{i})$$

<수식 1>

위 수식 1 에서 i 는 Rule 의 위치, R 은 총 Rule 의 수이며, f(i)는 i 번째 Rule 의 사용 빈도를 나타낸다.

다음은 way locking, entire locking 과 non-locking 의 경우에 대한 수식을 모델링 한다.

$$Pt = (Aclr \cdot Tc + Acr \cdot \frac{Acr}{A-Acr} \cdot Tc + Acr \cdot (1-\frac{Acr}{A-Acr}) \cdot (Tc+Tl) + Acout \cdot (Tc+Tl) + (N-1) \cdot n \cdot O)/A$$

<수식 2>

$$Pt = (Aclr \cdot Tc + Acout \cdot Ma + (N-1) \cdot W \cdot O)/A$$

<수식 3>

$$Pt = (Ac/A) \cdot Tc + (1-Ac/A) \cdot (Tc+Tl)$$

<수식 4>

위 수식 2,3,4 는 각각 way locking, entire locking 과 non-locking 의 경우에 대한 수식을 모델링 한 것이다. 이 수식 들에서 A 는 Rule access 총 횟수 이다.

$$Aclr = \sum_{i=1}^{nw} f(i) \cdot A$$

$$Acr = \sum_{i=nw+1}^{Ww} f(i) \cdot A$$

$$Aclr = \sum_{i=Ww+1}^{R} f(i) \cdot A$$

<수식 5>

수식 5 는 다음과 같은 각각의 영역 접근 수를 나타낸다.

Aclr 은 locking cache 영역, Acr 은 locking 안된 cache 영역, Acout 은 memory 영역, 그리고 Ac 는 전 cache

영역을 접근하는 횟수를 나타낸다.

또한, Pt 는 Rule 을 한 번 접근하는데 걸리는 평균 시간이며, N 은 locking 횟수, w 는 한 way 에 들어갈 수 있는 Rule 의 수, W 는 캐쉬가 가지고 있는 way 의 수, n 은 way 의 수를 나타낸다. Constant 로서 Ma 는 memory access time(64ns), Tl 은 line fill time(69ns), Tc 는 cache access time(4.76ns), 그리고 O 는 한 way 를 locking 하는데 걸리는 overhead time(768ns)이다. 괄호 안의 nanosecond 수치는 Motorola MPC8260 network processor 를 참조로 실험한 값을 나타낸다.

위 수식에서 사용되는 cache hit ratio 는 두 가지로서 locking 을 하지 않았을 때(수식 6)와 locking 을 했을 때(수식 7)로 나눌 수 있다.

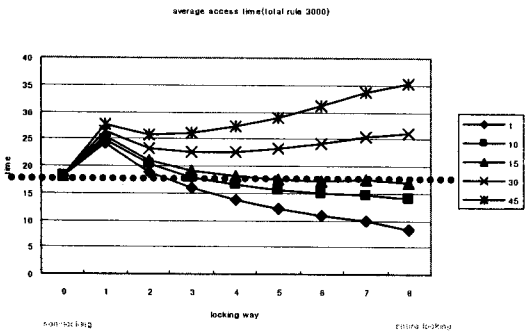
$$H = (Aclr + Acr)/A = Ac/A$$

<수식 6>

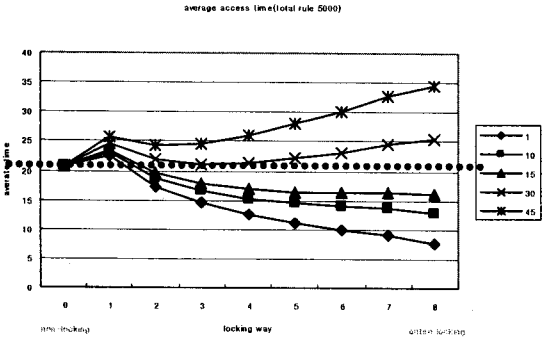
$$H' = Acr/(A - Aclr)$$

<수식 7>

3. 시뮬레이션 결과 및 분석



<그림 1>



<그림 2>

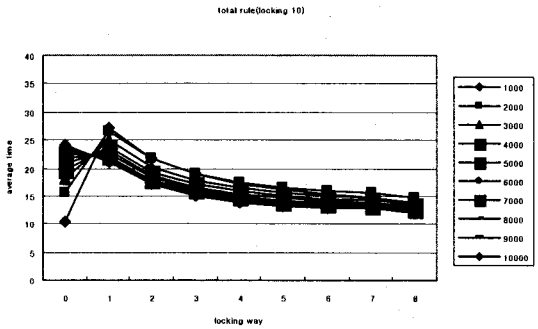
그림 1 과 그림 2 는 locking 주기와 locking way 를 변화 시켜가며, 한 Rule 당 평균 접근 시간을 구한 것이다. 그림 1 은 Total Rule 3,000 개일 때, 그림 2 는 Total Rule 이 5,000 개일 때를 Simulation 한 결과 이다.

Locking way 의 0 는 data-cache enable 상태에서 locking mechanism 을 전혀 적용하지 않은 상태로 성능의 기준이 된다. 그래프 상에서 이 지점보다 낮은 위치는 성능 향상 지점으로, 높은 위치는 성능 저하 지점으로 판단할 수 있다. Locking way 8 은 entire locking 으로 data-cache 전체를 locking 하는 것이다. Way locking 과의 차이는 entire locking 의 경우 캐쉬 외 영역을 cache-inhibited region 으로 인식한다는 것이다. 이 경우 locking 되어 있는 영역내의 데이터를 제외하고는 모든 데이터 접근은 직접 메모리로부터 패치해 온다.

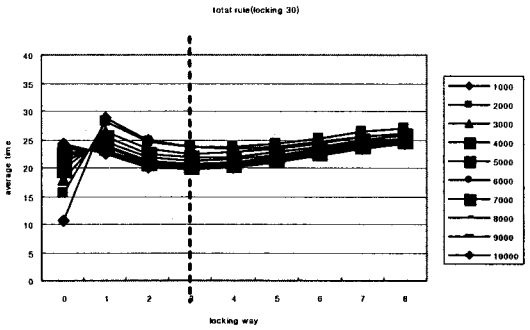
그림 1 과 그림 2 의 각 곡선은 locking 주기를 나타내며, 각 숫자는 단위 시간당 locking 한 횟수가 된다. 즉, 숫자가 높을수록 locking 주기는 빨라짐을 나타낸다.

먼저 그림 1 을 살펴보면, locking way 가 증가 할수록 Rule 평균 접근 시간이 감소함을 보이나, locking overhead 로 인해 locking 주기가 빨라 질수록 평균 접근 시간이 증가함을 보인다. 점선은 data-cache enable 상태에서 locking 을 하지 않았을 때를 기준으로 한 임계점을 나타낸다.

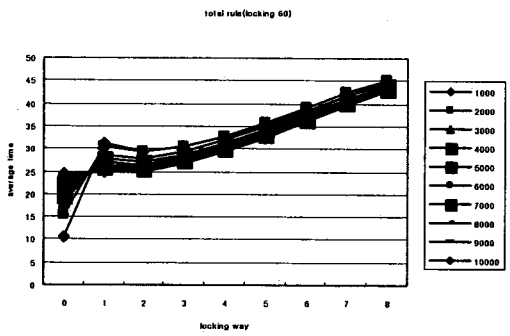
그림 1 과 그림 2 의 차이는 Total Rule 의 변화로서 그림 2 가 더 많은 Rule 을 가지고 Simulation 한 결과이다. Total Rule 이 증가함에 있어 locking 을 하지 않았을 경우, 평균 접근 시간이 증가함을 볼 수 있는 반면 locking 했을 경우 평균 접근 시간이 감소함을 볼 수 있다.



<그림 3>



<그림 4>



<그림 5>

그림 3 은 에서 각 곡선은 Total Rule 을 나타낸 것으로, 위에서 언급한 Total Rule 이 증가함에 따라 평균 접근 시간이 감소함을 보여준다.

그림 3,4,5 는 같은 조건하에서 locking 주기를 변화시켜가며 Simulation 한 결과 그래프로서 locking 주기가 어느 선을 넘어서면 기준 점인 non-locking 보다 평균 접근 시간이 증가함을 보여주고 있다.

그림 4 에서 보여주는 곡선은 locking way 증가와 locking overhead 의 상관 관계로 인해 최소 평균 접근 시간이 생기는 지점을 보여준다. 즉, 예로서 그래프의 한 곡선을 든다면, Total Rule 이 10,000 개(전체 캐쉬에 들어갈 수 있는 Rule 의 수는 8way * 64 개)이고 way 를 3 개 locking 했을 시 최적의 시스템 성능을 나타낼 수 있다.

4. 결론

본 논문에서는 Zipf's law 의 분포를 갖는 네트워크 패킷의 분포상황하에서 캐쉬 Size 와 Rule 의 개수를 고려할 때, 데이터 캐쉬 잠금 방법이 효과적으로 패킷 여과 시스템에 적용될 수 있음 알 수 있었다. 또한 단위 시간당 Rule 을 다시 캐쉬로 패치해오는 Overhead 를 고려할 때 빠른 데이터 캐쉬 잠금 주기는 오히려 패킷 여과 시스템의 성능 저하를 가져올 수 있음을 확인했다.

앞으로 Zipf's law 의 네트워크 분포상황 이외에서도 데이터 캐쉬 잠금 방법이 효용성 여부를 판단할 계획이다.

참고문헌

[1] R. Caceres, "Measurements of Wide-Area Internet Traffic," UCB/CSD 89/550, University of California, Berkeley, CA, December 1989.

[2] K. Claffy, "Internet Workload Characterization," Ph.D. Dissertation, University of California, San Diego, CA, June 1994.

[3] David C.Feldmeier " Improving gateway performance with a routing table cache" Massachusetts Institute of Technology Laboratory for Computer science IEEE ,1988

[4] Neeraj Gulati "Local Area Network Traffic Locality: Characteristics and Application"

[5] virgio Almeida , Azer Bestavros, Mark Crovella , "Characterizing Reference Locality in the WWW" , IEEE ,1996