End-to-End 네트워크 성능향상을 위한 자동화된 TCP Buffer Tuning 기술 개발*

류기철O(1) 심은숙(1) 김동균(1) 변태영(2) 석우진(3) 변옥환(3)
(1)경북대학교 (2)대구가톨릭대학교 (3)KISTI
{gcyooO, simy}@hotmail.com dongkyun@knu.ac.kr {wjseok25, ohbyeon}@kistil.re.kr

Development of Automatic TCP Buffer Tuning Technology for Improving the End-to-End Network Performance

Gichul YooO(1) Eunsuk Shim(1) Dongkyun Kim(1) TaeYoung Byun(2) Woojin Seok(3) Okhwan Byeon(3)
(1) Kyungpook National University (2) Catholic University of Daegu (3) KISTI

요 약

기존 TCP 기술은 높은 대역폭(High-Bandwidth) 및 큰 전송지연(High Delay)을 가진 통신에는 적합하지 못하다. TCP 기술의 성능향상을 위한 방법으로 TCP 제어 알고리즘을 수정하는 방법과 TCP Tuning 방법이 있다. 본 논문에서는 TCP Buffer Tuning 기술에 초점을 맞춰 통신망 상황에 따른 응용프로그램별로 자동화된 Buffer Tuning 기법을 제공하는 기술을 제안한다. ATBT(Automatic TCP Buffer Tuning) 에서는 송신측의 Buffer 크기를 조절하여 성능향상을 나타냈고, DRS(Dynamic Right Sizing)에서는 수신측의 Buffer 크기를 조절하여 성능향상을 도모하였다. 본 논문에서는 ATBT와 DRS의 장점을 접목하여 구현함으로써 보다 나은 성능향상을 나타내고 각 송·수신측의 모든 연결에 대해서는 Buffer를 공평하게 할당하여 메모리 사용의 효율을 높이고자 한다.

1. 서 론

BDP(Bandwidth-delay Product)가 높은 GRID와 같은 고속 데이터 통신망의 등장에도 불구하고 종단간(End-to-End) 데이터 전송의 성능향상을 얻을 수가 없다. 그래서 응용프로그램, 운영체제, 송·수신 측의 디스크 혹은 네트워크 Adapter, 네트워크 스위치와 라우터 등에서의 문제점을 파악하고 그 해결책을 얻으려는 노력을 많이 기울여 왔다. 하지만 성능향상을 얻지 못하는 원인 규명이 불분명 하여 실제 응용프로그램과 통신 서비스에 가장 밀접하게 연결되어 있는 TCP 부분의 성능향상을 위한 보완 작업을 집중적으로 많이 연구해 오고 있다.

TCP 성능향상을 위한 노력으로는 크게 TCP 제어 알고리즘을 수정하는 방법과 TCP Tuning 방법으로 나눌 수 있다. TCP 제어 알고리즘을 수정하는 방법은 TCP 혼잡제어(Congestion Control) 방법 혹은 흐름제어(Flow Control) 방법을 수정하여 실제 종단간 TCP 성능향상을 획득하려는 노력이다. 반면에, TCP Tuning 방법은 TCP 제어 알고리즘을 수정하지 않고 기존의 TCP 기술을 그대로 수용한다. 하지만 실제 TCP를 구동하는데 있어 변경 가능한 Parameters를 Network 상황에 맞게 조절함으로써 종단간 성능향상을 얻고자 한다. 그 대표적인 Tuning 기술로 TCP Buffer Tuning 기술을 들 수가 있다.

* 본 논문은 한국과학기술정보연구원 슈퍼컴퓨팅센터 위탁과제로 지원 되었습니다.

이 논문에서는 기존에 제시된 TCP Buffer Tuning 방법인 ATBT(Automatic TCP Buffer Tuning)와 DRS (Dynamic Right Sizing)의 장점을 이용한 Hybrid TCP Buffer Tuning 기술을 제안한다.

이후 본 논문 2장에서는 기존의 TCP Tuning 기술을 살펴보고, 3장에서는 End-to-End 네트워크 성능향상을 위한 Hybrid TCP Buffer Tuning 기술을 제안하고, 4장에서 구현방법을 살펴보고 마지막으로 5장에서 결론을 맺는다.

2. 기존의 TCP Tuning 기술

2.1 ATBT(Automatic TCP Buffer Tuning)

일반적인 TCP 연결은 64Kbytes와 같은 Default 값으로 고정된 작은 Buffer 크기를 가지고 있다. 사실상 BDP가 큼에도 불구하고 송신측에서 전송하는 데이터 량이 적어서 종단간(End-to-End) 성능향상을 얻기가 힘들고 지연시간도 크다. 만약 Default Buffer 크기를 크게 했을 경우, 단일 시스템 위에서 응용프로그램 수가 많아지면 결국 Buffer에 대한 경쟁이 생겨 시스템 메모리 남용이나 병목현상과 같은 문제점이 따르게 된다.

이러한 문제점을 해결하고자 Auto Tuning 방법이 제안되었다. Auto Tuning의 목표는 각 TCP 연결에게 어떠한 환경에라도 최상의 수행 능력을 줄 수 있고, 더욱 효과적으로 시스템 메모리를 사용하면서 메모리 낭비를 막자는데 있다. ATBT는 BDP의 동적인 변화에 영향을 받

게 되고, 네트워크 상황과 시스템 메모리의 유효성을 기반으로 한다. ATBT는 크게 송신측과 수신측으로 나눠서 생각할 수 있다. 송신측에서의 Send Socket Buffer는 Network-based target, Fair Share of Memory, Memory Threshold의 세가지 알고리즘에 의해서 정해진다.

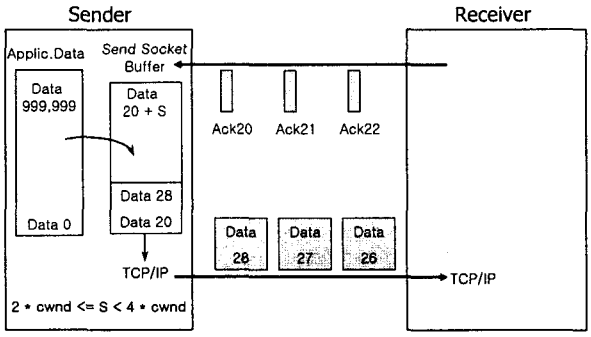

그림 1 ATBT 송신측 Buffer Tuning

그림 1에서와 같이 ATBT는 송신측 Buffer 크기를 CWND의 2배에서 4배의 크기로 미리 확보해 둔다. 반면에 수신측 Buffer는 Operating System의 최대 크기로 설정하게 된다[1].

2.2 DRS(Dynamic Right Sizing)

고성능의 계산적인 GRID와 그 외 대역폭과 관련된 응용에서 TCP와 관련한 흐름제어와 혼잡제어 문제가 있다. 이를 해결하기 위해 관련 연구가들은 수동적으로 최적의 Buffer 크기를 유지하고자 하였다. 그 결과로 GRID 컴퓨팅 환경인 WAN(wide-area network)에서 어느 정도의 성능 향상은 얻을 수 있었다. 하지만 이러한 Tuning 프로세스가 각 연결마다 RTT와 Bottleneck link의 대역폭을 계산해야 하므로 사용자나 보통의 개발자들에게는 어렵다는 문제점이 있다[2].

현재는 Buffer 크기를 적당하게 Tuning 하기 위해서 iperf, nettimer, netspec, nettest, pchar, pipechar와 같은 RTT와 Bottleneck link의 대역폭을 결정하는 툴을 사용한다. 하지만 이러한 툴 역시 클라이언트 API를 제공하지 않고 특정 수준의 사용자만이 사용 가능하다는 문제를 안고 있다.

이에 따라 자동으로 수신측 Buffer를 Tuning 하고자 DRS(Dynamic Right Sizing) 방법이 제안되었다[3]. 간단하게 말하자면, DRS는 수신측이 송신측의 CWND를 예측하여 수신측의 Window Advertisement의 크기를 동적으로 변화시킨다. 이는 연결이 설정되는 순간만 동적으로 Buffer를 Tuning하는 것이 아니라 연결의 life-time 동안에 계속적으로 변한다.

DRS 방법을 위해, 수신측의 가용한 대역폭과 RTT를 결정해야 한다. 평균 가용 대역폭은 수신된 바이트 수를 RTT로 나누어 얻을 수 있다. 대역폭을 예측하는 방법과는 달리 RTT는 원래 송신측이 데이터를 보내고 그 데이터에 대한 ACK를 수신한 시간으로 계산된다. 하지만 수신측에서는 네트워크 패킷을 보내고 ACK를 받는 동작의 트래픽을 조사해 보지 않고는 알 수 없다. 즉, 예상 RTT를 측정하기 위해서 그림 2 에서와 같이 하나의 패킷에 대한 ACK를 보낸 후에 그 ACK에 따른 데이터가 도착한 시간으로 결정한다.

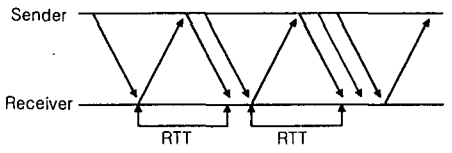

그림 2 수신측 RTT 예측

예측된 가용 대역폭과 RTT의 곱을 BDP의 값으로 한다. 따라서 Slow-start 동안에 FWND 의 크기는 2 배의 BDP가 되고 Congestion Avoidance 구간에서는 선형적으로 증가하게 된다.

2.3 SABT(Scalable Automatic Buffer Tuning)

TCP Send Socket Buffer를 자동으로 Tuning 하고자 하는 방법이다. 특히 송신 호스트에서 많은 TCP 연결설정 요구를 동시에 수락했을 경우, 각 TCP 연결마다 그 특성을 고려하여 Send Socket Buffer 크기를 설정한다.

ATBT에서는 각 TCP 연결에 할당되는 Buffer 크기가 현재 사용되는 CWND의 값에 따라 결정된다. 모든 TCP 연결들의 전체 요구되는 Buffer 크기가 Send Socket Buffer 크기보다 클 경우, Send Socket Buffer는 각 TCP 연결에 MAX-MIN Fairness 정책에 따라 할당된다. 즉 우선적으로 Buffer를 동등하게 모든 TCP 연결에 할당한다. 큰 Buffer를 요구하지 않는 연결이 존재할 경우, 사용되지 않는 부분을 할당된 크기 보다 더 많이 요구하는 연결에게 재할당 한다.

이런 상황에서, 만약 ATBT에서 TCP가 갑자기 CWND 값을 변형할 경우 할당된 Buffer 크기는 때때로 연결이 실제로 요구하는 것보다 더 작게 설정될 수 있으며 Buffer 크기 할당이 너무 자주 변경될 수 있는 문제점을 가지고 있다. 또한 통신망 대역폭이 클 때 CWND의 Oscillation이 크게 일어나 결국 할당된 Buffer 크기의 Oscillation을 초래할 수 있다. 반면 SABT는 잔여 Buffer 크기를 연결별로 요구되는 량이 다를 경우 요구량에 비례해서 재 할당하게 하는 방법을 제공하고 있다[4].

3. Hybrid TCP Buffer Tuning

Hybrid TCP Buffer Tuning 방법은 기존에 제시된 ATBT의 송신측 Buffer 할당 방법과 DRS의 수신측 Buffer 할당 방법의 장점을 이용한다. 따라서 송·수신 Buffer 할당 기법을 다음과 같이 요약할 수 있다.

3.1 송신측에서의 Buffer 할당

ATBT에서와 마찬가지로 송신측 Buffer 할당에 있어서 CWND의 2배에서 4배 사이의 Buffer 공간을 미리 확보해 놓음으로써 응용프로그램에서 내려받은 데이터를 CWND 값의 변화에 따라 신속히 전송할 수 있다. DRS에서는 수신측에서 송신측의 CWND를 예측하여 CWND의 2배만큼의 Buffer를 수신측에 설정한다. 이 설정된 정보를 송신측에 전달하면서 결국 그 값으로 송신측의 Buffer 크기를 결정하려고 한다. 하지만 수신측의 CWND 예측의 정확성에 전적으로 성능이 영향을 받기 때문에 본 논문에서는 ATBT 형태의 송신측 Buffer 할당 방법을 채택한다.

또한 Buffer 할당에 있어 TCP 연결 수가 많아 졌을 경우 각 TCP 연결에 공정하게 Buffer를 할당 받을 수 있는 방법을 제공해야 한다. ATBT에서 사용하는 Fair Share 알고리즘과는 달리 요구되는 Buffer 요구량에 따라 잔여 Buffer 할당을 좀더 효율적으로 적용할 수 있는 방법을 채택한다. 따라서 SABT에서 사용하고 있는 Fair Share 기법을 적용하여 Weighted Fair Sharing을 가능하게 한다[4].

3.2 수신측에서의 Buffer 할당

ATBT에서 수신측 Buffer는 단순히 최대 값으로 설정되어 있다고 가정한다. 또한 수신측에서의 TCP 연결 개수에는 전혀 무관하게 동작하고 있다. 예를 들면 수신측에 TCP 연결이 많이 존재 할 경우 결국 그 TCP 연결간에도 제한된 크기의 Buffer를 공정하게 배분해야 하는 문제점을 가지고 있다.

반면에 DRS에서는 수신측 Buffer를 단순히 예측된 CWND의 2배로 설정하게 되지만 RTT 값에 의존하여 CWND 예측을 하기 때문에 정확성이 가장 큰 문제이다. 또한 여러 TCP 연결간에 공정하게 Buffer를 할당하는 방법이 제시되어 있지 않다. 따라서 본 논문에서는 SABT에서 비록 송신측에서의 Buffer 할당에 Weighted Fair Sharing 기법을 적용하였지만 수신측 Buffer 할당에 있어서도 동일한 기법을 적용하도록 한다.

4. 구현

본 논문에서 제안한 Hybrid TCP Buffer Tuning은 종단간의 송·수신측 Buffer Tuning 기법을 동시에 구현한다. 구현은 Linux Kernel 2.4.20 에서 구현 중에 있다. TCP 소켓에서는 IP 계층으로부터 sk_buff 형태로 버퍼를 넘겨받아서 프로토콜에 관련된 연산을 처리한 후 데이터를 전송한다. /proc/sys/net/ip4에는 송·수신측 Buffer를 위해 예약된 메모리를 나타내는 TCP 변수인 tcp_wmem와 tcp_rmem이 있다. 각 변수는 min, default, max 값이 설정되어 있다. 송신측 Buffer 크기를 Tuning하기 위해서 sndmem은 CWND의 2배와 4배 사이로 설정하고 tcp_wmem은 min값으로 설정한다. 따라서, 송신측 Buffer(sk->sndbuf) 크기는 sndmem와 tcp_wmem의 최소값으로 결정한다. 그리고 수신측 Buffer (sk->rcvbuf) 크기를 Tuning하기 위해서 rcvmem은 추정된 CWND 2배로 설정하고 tcp_rmem은 min 값으로 설정한다. 따라서 수신측 Buffer 크기는 rcvmem와 tcp_rmem의 최소값으로 결정한다. 수신측의 tcp_rmem이 max값을 넘지 않을 동안은 연결마다 동적으로 할당된 Buffer 사이즈를 유지한다. 그러다가 새로운 연결이 발생하여 tcp_rmem의 max값을 초과할 때에는 fair sharing을 위해 SABT에서 언급했던 Fairness 정책을 사용한다. 구현된 시스템은 기존의 시스템과 비교하여 종단간의 전송율, 혼잡제어 윈도우 크기, 버퍼 사용량에 대해 향상된 성능을 보이고자 한다.

5. 결론

TCP Buffer Tuning 기술을 통해 슈퍼컴퓨팅 등 첨단 컴퓨팅 환경에서 동작하는 응용프로그램에게 최적의 성능향상을 도모하고자 한다.

송신측 Buffer를 2배에서 4배 크기로 확보해 두는 기술을 송신측에 구현하고 수신측에서는 송신측의 CWND를 예측하여 Window Advertisement의 크기를 동적으로 변화하도록 하였다. 또한, 모든 TCP 연결들의 전체 요구되는 Buffer 크기를 연결별로 요구되는 량이 다를 경우 요구량에 비례해서 재 할당하게 하는 방법을 제안 하였다. 본 논문에서는 송·수신측의 Buffer를 모두 고려함으로써 보다 나은 TCP 성능향상을 도모하고자 한다.

참 고 문 헌

[1] J.Semke, J.Mahdavi and M.Mathis, "Automatic TCP Buffer Tuning," ACM SIGCOMM 1998, vol. 28, no.4 1998

[2] Eric Weigle and Wu-chun Feng, "Dynamic Right-Sizing:A Simulation Study," IEEE ICCCN, 2001.

[3] M. Fisk and W. Feng. "Dynamic Right Sizing in TCP," In Proceeding of the Los Alamos Computer Science Institute Symposium, Oct 2001. LA-UR 01-5460

[4] Takahiro Matsuo, Go Hasegawa and Masayuki Murata, "Scalable Automatic Buffer Tuning to Provide High Performance and Fair Service for TCP Connection," In Proceedings of IEEE INET 2000, July 2000