분산클라우드 환경에서 마이크로 데이터센터간 자료공유 알고리즘★

김현철*

요 약

현재의 ICT 인프라(인터넷과 서버/Client 연동)는 다양한 장치, 서비스, 비즈니스 및 기술 진화에 따른 신속한 대응에 어려움을 겪고 있다. 클라우드 컴퓨팅(Cloud Computing)은 구름 같은 네트워크 환경에서 원하는 작업을 요청하여 실행한다는 데서 기원하였으며, 인터넷 기술을 활용하여 IT 자원을 서비스로 제공하는 컴퓨팅을 뜻하고 오늘날 IT 트렌드의 하나로 가장 주목 받고 있다. 이러한 분산클라우드 환경에서는 네트워크 및 컴퓨팅 자원에 대한 통합 관리 체계를 통하여 관리 비용 증가 문제를 원천적으로 해결하고 분산된 마이크로 데이터센터(Micro DC(Data Center))를 통하여 코어 네트워크 트래픽 폭증 문제를 해결하여 비용 절감 효과를 높일 수 있다. 그러나 기존의 Flooding 방식은 인접한 모든 DC들에게 전송하기 때문에 많은 트래픽을 유발 할 수 있다. 이를 위해 Restricted Path Flooding 알고리즘이 제안되었으나 대규모 네트워크에서는 여전히 트래픽을 발생할 수 있는 단점이 있어서 본 논문에서는 홉수 제한을 통하여 이를 개선한 Lightweight Path Flooding 알고리즘을 제안하였다.

A Data Sharing Algorithm of Micro Data Center in Distributed Cloud Networks

Hyuncheol Kim*

ABSTRACT

Current ICT(Information & Communication Technology) infrastructures (Internet and server/client communicatio n) are struggling for a wide variety of devices, services, and business and technology evolution. Cloud computing or iginated simply to request and execute the desired operation from the network of clouds. It means that an IT resour ce that provides a service using the Internet technology. It is getting the most attention in today's IT trends. In the distributed cloud environments, management costs for the network and computing resources are solved fundamentall y through the integrated management system. It can increase the cost savings to solve the traffic explosion problem of core network via a distributed Micro DC. However, traditional flooding methods may cause a lot of traffic due to transfer to all the neighbor DCs. Restricted Path Flooding algorithms have been proposed for this purpose. In large networks, there is still the disadvantage that may occur traffic. In this paper, we developed Lightweight Path Floodi ng algorithm to improve existing flooding algorithm using hop count restriction..

Key words : Cloud Computing, Micro Data Center, Data Sharing

접수일(2015년 3월 12일), 게재확정일(2015년 3월 30일)

★ 이 논문은 2014년도 남서울대학교 학술연구비 지원에 의해 연구되었음.

* 남서울대학교 컴퓨터학과 교수

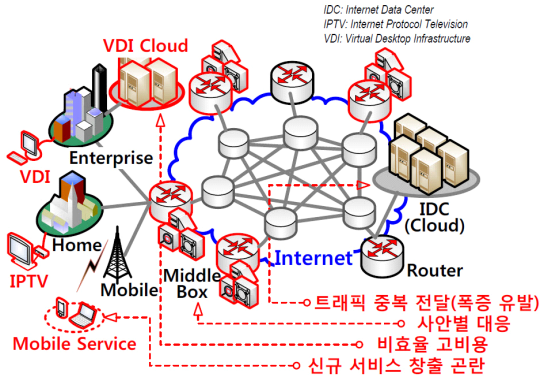
1. 서 론

현재의 네트워크는 고품질, 대규모, 대용량 서비스 제공이 어려운 구조이며 트래픽 용량 증가와 통신 사업자 매출의 불일치로 인해 트래픽 증가 대비 수익이 감소하는 현상 발생하여 사업자(통신사업자와 콘텐츠 사업자) 간에 악순환 지속되고 있다. 특히 웹기반의 비디오 트래픽 급증과 클라우드 서비스 확대에 따른 트래픽 빅뱅(약 연간 2배 이상 급증)으로 현재의 네트워크 구조로는 신규 서비스 창출이 어려운 상황이다 [1][2]. 이러한 투자-수익간 불균형 문제를 해결하기 위한 통신사업자를 중심으로 하드웨어 기반 기존 네트워크 장비에 대한 불만 확산되고 있으며 네트워크 가상화, 네트워크-클라우드 통합 관리, 응용 기반 네트워크 등에 대한 기술 개발 요구가 증대되고 있다.

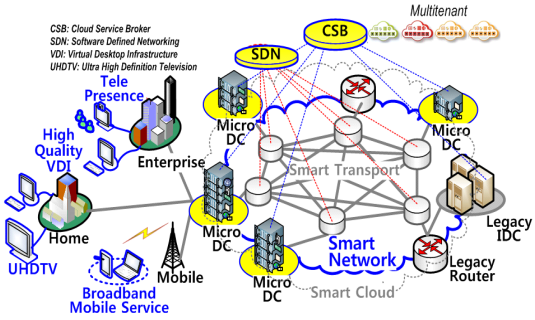
클라우드 컴퓨팅 사용자는 필요에 따라 무한정의 컴퓨팅 자원을 제공 받을 수 있으며 단지 사용자는 시스템 자원에 대한 요구가 증가함에 따라 시스템 자원을 증가시키면 된다. 또한 필요에 따라 짧은 시간 단위로 사용하고 비용을 지불하면 되고 필요가 사라지면 자원을 더 사용하지 않을 수 있다 [3]. 이러한 추세를 반영하여 (그림 1)에서와 같이 네트워킹 된 ICT (Information & Communication Technology) 자원 및 기능이 대규모로 가상화되고 통합 제어되며 개방을 통해 사업자와 사용자에게 고품질 저비용으로 혁신적 서비스를 제공하는 개방형 플랫폼까지 등장하게 되었다 [4][5].

그러나 데이터센터는 IoT(Internet of Things)의 대규모 데이터 실시간 처리로 작업량 증가, 새로운 보안, 용량 및 분석에 직면하게 되었다. 이러한 IoT를 위한 막대한 데이터와 접속은 (그림 2)에서와 같은 분산 데이터센터(Distributed DC(Data Center))를 기반으로 하는 네트워크 구성을 가속화 시켰다.

본 논문에서는 Micro DC간의 신속한 데이터 공유를 위해 기존의 Restricted Path Flooding 알고리즘을 개선한 Lightweight Path Flooding 알고리즘을 제안하였다.



(그림 1) 대규모 IDC 기반 클라우드 네트워크



(그림 2) Micro DC 기반 분산클라우드 네트워크

본 논문의 구성은 다음과 같다. 2장에서는 데이터 공유를 위한 분산 프로토콜 관련 선행 기술 조사 및 분석을 수행하였다. 3장에서는 제안된 Lightweight Path Flooding 알고리즘의 개념과 동작방식을 설명하였다. 마지막으로 4장에서는 제안된 Lightweight Path Flooding 알고리즘의 성능을 분석하였다.

2. 분산 프로토콜

분산 전송 프로토콜은 대체적으로 어플리케이션 레벨에서 사용되고, 이는 사용자에게 데이터 공유에 편리함을 준다. 기존에 파일을 공유하기 위해서는 서버-클라이언트 형태로 되어 있는 FTP를 주로 사용하였다. 이는 데이터를 공유하는 서버를 알아야지만, 사용자가 원하는 데이터를 전송 받을 수 있다 [6].

분산 전송 프로토콜의 경우 (표 1)에서와 같이 파일을 공유하는 서버나 파일을 전송받는 클라이언트로 나뉘지 않고 각각의 호스트가 피어(Peer)라 불리며 서버와 클라이언트 역할을 동시에 담당한다. 따라서 데이터 공유 시 사용자가 쉽게 데이터 자원을 찾을 수 있고, 여러 피어로부터 데이터를 공유 받기 때문에 더 빠르게 데이터 자원을 얻을 수 있다 [7][8].

일반적으로 공유 자원을 검색 방법으로는 두 가지 방식이 사용된다. 하나는 무작위 검색 방법이고 다른 하나는 정해진 룰에 의해 정보를 얻고 얻어진 정보에 의한 검색 방법이다. 정보에 의한 검색 방법은 질의 메시지는 사용자가 어떠한 데이터를 요청하기 위해 현재의 상황을 파악하기 위해 사용되는 메시지로 자신과 인접한 피어들에게 보내는 데이터 메시지이다. 질의 메시지를 전송하기 위해 사용하는 방식은 Flooding 기반 방식과 non Flooding 기반 방식으로 나눌 수 있다.

<표 1> 분산프로토콜 비교

Peer-to-Peer				
1. 자원은 피어 간에 공유된다. 2. 자원은 다른 피어로부터 직접 접근 가능하다. 3. 피어는 자원을 제공하는 제공자인 동시에 자원을 요구하는 요구자 이다.				
Unstructured P2P				Structured P2P
1세대		2세대		
Centralized P2P	Pure P2P	Hybrid P2P		DHT-Based
1. 서비스 제공을 위한 Central Entity 필요 2. Central Entity는 인덱스/그룹 DB 역활	1. 모든 터미널이 분산되어 있기 때문에 어떤 하나의 터미널이 오류가 나더라도 전체 네트워크에 문제가 없음 2. Central Entity 없음	1. 모든 터미널이 분산 2. 어떤 하나의 터미널이 오류가 나더라도 전체 네트워크에 문제가 없음 3. 동적인 Central Entity 가 필요	1. 검색 기능이 존재하지 않음 2. 필요한 데이터의 위치를 사용자가 수동으로 찾아야 함. 3. 데이터자원의 공유가 빈번 할수록 데이터 전송 속도가 빨라짐	1. 모든 터미널이 분산되어 있기 때문에 어떤 하나의 터미널이 제거 되거나 오류가 나더라도 전체 네트워크에 문제가 없음 2. Central Entity 없음 3. Overlay 상 피어간 연결이 고정됨
Napster	Gnutella 0.4 , Freenet	Gnutella 0.6 , JXTA,	Bittorrent	Chord, CAN, FastTrack

3. Lightweight Path Flooding 알고리즘

기존 방법의 경우, 정보 데이터나 Ping, Pong 메시지로 인해서 불필요한 데이터가 많이 발생하게 된다. 또한 질의 메시지를 생성할 경우, 이는 Flooding 방식을 사용하기 때문에 중복된 데이터가 계속 발생하므로 네트워크 내에 과부하가 생길 수 있다. 이는 혼잡을 발생시키게 되고, 실제 데이터를 전송할 때에 어려움을 줄 수 있다.

3.1 기존방식의 문제점

기존 Flooding 알고리즘은 어떤 DC가 질의 메시지를 보내려고 하거나 질의 메시지를 받은 후, 다른 피어 DC들에게 전달 하고자 할 때 인접한 모든 피어들에게 전송하는 방식을 사용하였다. 이러한 방식은 대규모 네트워크에서 정보를 빠르게 배포할 수 있는 장점을 제공하지만 Flooding방식은 네트워크 내에 대규모의 트래픽을 발생시킨다. 이는 네트워크 전체에 과부하를 줄 뿐만 아니라 병목 현상과 같은 문제를 발생시킨다.

Flooding의 또 다른 문제점은 Flooding에 의해 전송되는 질의 메시지가 높은 연결성을 가진 DC에 중복으로 전송될 확률이 높다는 것이다. 중복으로 전송되는 경우 이에 대해 또 Flooding을 하게 되면 중복된 질의 메시지를 받는 DC들은 계속 증가하게 된다.

또한 계속 비슷한 DC들에게만 전송됨에 따라서 목적 DC를 찾아내기 힘들어지는 문제점을 야기한다. 이를 막기 위해서 Flooding-Style의 검색방안을 사용하는 공유방식은 동일한 질의 메시지를 받을 때 메시지를 폐기하고 다시 Flooding 하지 않는 방안을 사용한다. 하지만 Flooding을 하지 않는다고 할지라도 중복 메시지는 이미 과도하게 발생하기 때문에 그 효용성이 낮다.

3.2 Lightweight Path Flooding 알고리즘

임의의 DC에서 데이터 자원을 요청하기 위해 질의 메시지가 발생되었다고 가정하면 질의 메시지를 보낼 DC는 인접 DC에게 자신이 생성한 질의 메시지를 Flooding 하게 되는 데, 우선 네트워크 환경에 맞는 TTL 값을 설정하고 질의 내용과 인접 DC 주소를 포함한 질의 메시지를 만들고 이를 Flooding 한다.

질의 메시지를 수신한 DC는 해당 데이터를 가지고

있는지를 확인한 후에, 만일 있다면 이에 대한 응답을 보내게 되고, 없다면 질의 메시지 내의 TTL 값이 1보다 큰지를 확인해서 큰 값을 가질 경우와 해당 질의 메시지가 이전에도 수신되었던 것인지를 확인해서 아닐 경우에만 질의 메시지를 Flooding을 한다. 이때 메시지 내에 저장되어 있는 S_address(이 메시지가 전송되었던 Peer DC들의 주소)와 자신과 인접해 있는 Peer DC들의 주소를 비교해서 같은 주소가 있을 경우에는 해당 주소를 자신이 보내야 할 주소 목록에서 지운다.

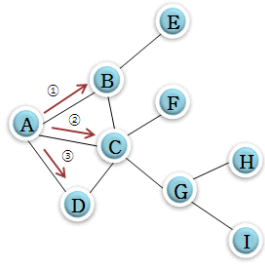
또한 모든 주소 목록에 대해서 비교를 해서 자신이 질의 메시지를 보낼 주소 목록을 최적화 한 후, 자신의 주소목록을 이전의 S_address 목록에 추가하여 메시지를 보낸다. 이로써 최대한 질의 메시지가 같은 DC에 중복해서 전송되는 경우를 최소화한다. 또한 홉이 2 이상 되는 DC에 대한 정보를 S_address에서 제거함으로써 S_address가 불필요하게 커지는 것을 방지하고 네트워크 트래픽을 줄일 수 있게 된다. (그림 3)은 Lightweight Path Flooding 알고리즘의 슈도코드이다.

예를 들어, (그림 4) 네트워크에서 DC A가 데이터 요청을 위해서 질의 메시지를 전송한다고 하자. (그림 4)에서 ①, ②, 그리고 ③은 DC A로부터 전송되는 메시지를 의미한다. (그림 5)는 DC A의 주소 목록이고 (그림 6)은 질의 메시지에 들어가는 S_address 목록이다.

```
address  ←  IP address_list adjoin to own
data  ←  requested data name or hash values
S_address  ←  IP address list in transmitted query
message

Send Query ( income message )
1  IF TTL value in message > 1 && data in message is
     new
2     For S_address in message ← start point
3        to S_address in message ← end point
4           IF (compare s_address in message with
address_list)
5              eliminate relevant address in address_list
6           End IF
7     End for
8     add address_list to S_address
9  Else
10    Remove this message
11 End if
12 eliminate |hop >2| address in address_list
13 make packet with data & S_address
14 Send (query message)
```

(그림 3) Lightweight Path Flooding 슈도코드



(그림 4) 예제 네트워크

Peer
B
C
D

(그림 5) DC A에서의 주소

Peer
A
B
C
D

(그림 6) DC A에서의 S_address

질의 메시지를 수신한 DC B, C, D에 DC A가 요청하는 데이터가 없다고 한다면, DC들은 다시 질의 메시지를 Flooding을 한다. (그림 7)은 DC C의 인접 DC 주소목록이고, (그림 8)은 받은 질의 메시지 내의 S_address 목록을 토대로 작성된 DC C의 S_address 초기 목록이다.

Peer
A
B
D
F
G

(그림 7) DC A에서의 주소

Peer
A
B
C
D
F
G

(그림 8) DC A에서의 초기 S_address

DC C의 인접 DC는 A, B, D, F, G가 있지만 이미 받은 S_address를 통해 A, B, D는 전송된 것을 알 수 있다. 따라서 DC C는 인접한 DC 중 F, G에게만 질의 메시지를 전송한다. 또한 DC F, G와는 홉수가 2 이상인 DC A, B, D 주소는 S_address에서 삭제하고 전달된다. 따라서 Lightweight Path Flooding 방식은 거대한 네트워크에서도 전달 지연을 최소화 할 수 있다.

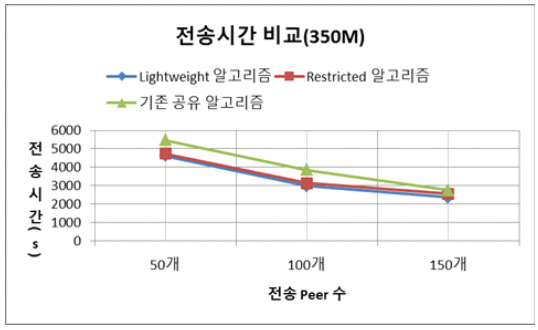
Peer
C
F
G

(그림 9) DC A에서의 최종 S_address

4. 성능평가

성능 측정은 600Mbytes, 350Mbytes, 100Mbytes, 1Mbytes의 파일을 각각 50, 100, 150개의 피어로부터 전송 받을 때, 걸린 시간과 속도를 측정하였다. 파일 전송 시간은 파일을 다운 받기 시작한 시간부터 파일의 무결성 유무를 판단하는 시간까지를 측정했으며, 전송 속도는 파일을 다운이 완결 될 때까지의 시간을 이용하였다.

(그림 10) 전송시간 비교 (600M)



(그림 11) 전송시간 비교 (350M)

(그림 10)에서와 같이 600Mbytes를 전송하는 경우, 제안한 방식은 기존의 방식보다는 최대 40%, Restricted Flooding 알고리즘 보다는 약 4% 정도의 성능향상을 제공하였다.

또한 (그림 11)에서와 같이 350Mbytes를 전송하는 경우, 제안한 방식은 기존의 방식보다는 최대 19%, Restricted Flooding 알고리즘 보다는 약 2.8% 정도의 성능향상을 제공하였다.

4. 결 론

클라우드 컴퓨팅과 관련된 기술들은 계속 발전하고 있으며, 짧은 기간 안에 수많은 컴퓨터를 연결해 컴퓨터의 계산 능력을 극대화할 수 있을 것으로 보인다. 그러나 클라우드 컴퓨팅이 성공적으로 수행되기 위해서는 분산된 시스템들 간에 자원의 이동이 원활하게 이루어져야 하는데 네트워크 선로상의 문제나 호스트

IP의 문제로 이를 방해받는 경우가 생기게 된다.

특히 대용량 데이터를 전송할 때에 갑작스레 전송이 중단되면 네트워크 자원의 손실이 크다. 본 연구에서 제안한 데이터 공유 프로그램을 이용하면 대용량 데이터 전송이 빈번하게 일어나는 분산 클라우드 네트워크에서 보다 안정적이고 신뢰성 있는 데이터 공유가 가능하다. 또한 고속 분산데이터 전송 프로토콜을 통한 빠른 데이터 검색 및 전송으로 데이터 자원 공유의 편리성을 높일 수 있다.

참고문헌

[1] Wang En Dong, et.al,, “QoS-Oriented Monitoring Model of Cloud Computing Resources Availability,” IEEE International Conference on Intelligent Computing and Intelligent Systems, Vol. 3, pp. 642-646, 2013.

[2] Amanatullah, Y., et. al,, “Toward cloud computing reference architecture: Cloud service management perspective,” IEEE International Conference on ICT for Smart Society, pp. 1-4, 2013.

[3] Malik, S,. et.al., "Cooperative cloud computing in research and academic environment using Virtual Cloud," IEEE International Conference on Emerging Technologies, pp. 1-7, 2012.

[4] Jadeja, Y., Modi, K., “Cloud computing - concepts, architecture and challenges,” IEEE International Conference on Computing, Electronics and Electrical Technologies, pp. 877-880, 2012.

[5] Mollah, M.B., et.al,, “Next generation of computing through cloud computing technology,” IEEE International Conference on Electrical & Computer Engineering, pp. 1-6, 2012.

[6] Shuai Zhang, et.al,, “The comparison between cloud computing and grid computing,” IEEE International Conference on Computer Application and System Modeling, pp. 11-72, 2010.

[7] Panneerselvam, J., et.al,, "An Investigation of the Effect of Cloud Computing on Network Management," High Performance Computing and Communication, pp. 1794-1799, 2012.

[8] Islam, S.S., et.al,, “Cloud computing for future generation of computing technology,” IEEE International Conference on Cyber Technology in Automation, Control, and Intelligent Systems, pp. 129-134, 2012

[저 자 소 개]

김 현 철 (Hyuncheol Kim)

1990년 2월 성균관대학교 학사
1992년 2월 성균관대학교 석사
2005년 8월 성균관대학교 박사
2006년 9월 ~ 현재 남서울대학교 컴퓨터학과 교수

email : hckim@nsu.ac.kr