Hadoop 기반 분산 컴퓨팅 환경에서 네트워크 I/O의 성능개선을 위한 TIPC의 적용과 분석

(Applying TIPC Protocol for Increasing Network Performance in Hadoop-based Distributed Computing Environment)

유 대 현 † 정 상 화 †† 김 태 훈 †††
(Dae Hyun Yoo) (Sang Hwa Chung) (Tae Hun Kim)

요 약 최근 인터넷 서비스 기반의 데이터는 대용량화되고 있으며 대용량 데이터를 효과적으로 처리할 수 있는 구글 플랫폼, Apache Hadoop과 같은 플랫폼 기술이 각광받고 있다. 이러한 플랫폼에서는 분산 프로그래밍을 위한 기법으로 MapReduce가 수행되며, 이 과정에서 각 태스크의 결과를 전달하기 위한 네트워크 I/O의 부하 문제가 발생한다. 본 논문에서는 구글 플랫폼, Hadoop과 같은 대규모 PC 클러스터 상의 분산 컴퓨팅 환경에서 네트워킹 부하를 경감하고 성능을 향상시키는 방안으로 TIPC(Transparent Inter-Process Communication)의 적용을 제안한다. TIPC는 경량화된 연결설정 및 스택 크기, 계층적 주소체계로 인해 TCP보다 가볍고 CPU 부하가 적은 장점을 가지고 있다. 본 논문에서는 Hadoop 기반 분산 컴퓨팅 환경의 특징을 분석하여 그와 유사한 실험환경을 모델화하고 다양한 프로토콜의 비교실험을 수행하였다. 실험결과 평균 전송률에서 CUBIC-TCP, SCTP와 비교해 TIPC의 성능이 가장 우수하였으며, TIPC는 CPU 점유율 측면에서 TCP와 비교해 최대 15%의 낮은 CPU 점유율을 보였다.

키워드 : 분산 컴퓨팅 환경, Hadoop, MapReduce, TIPC

Abstract Recently with increase of data in the Internet, platform technologies that can process huge data effectively such as Google platform and Hadoop are regarded as worthy of notice. In this kind of platform, there exist network I/O overheads to send task outputs due to the MapReduce operation which is a programming model to support parallel computation in the large cluster system. In this paper, we suggest applying of TIPC (Transparent Inter-Process Communication) protocol for reducing network I/O overheads and increasing network performance in the distributed computing environments. TIPC has a lightweight protocol stack and it spends relatively less CPU time than TCP because of its simple connection establishment and logical addressing. In this paper, we analyze main features of the Hadoop-based distributed computing system, and we build an experimental model which can be used for experiments to compare the performance of various protocols. In the experimental result, TIPC has a higher bandwidth and lower CPU overheads than other protocols.

Key words : Distributed Computing Environment, Hadoop, MapReduce, TIPC

· 이 논문은 2008년 교육과학기술부로부터 지원받아 수행된 연구임(지역거점연구단육성사업/차세대물류IT기술연구사업단)

† 정 회 원 : 삼성탈레스 S/W 2그룹
Daehyun81.yoo@samsung.com
†† 종신회원 : 부산대학교 컴퓨터공학과 교수
shchung@pusan.ac.kr
(Corresponding author임)
††† 학생회원 : 부산대학교 컴퓨터공학과
likeacat@pusan.ac.kr

논문접수 : 2009년 2월 23일
심사완료 : 2009년 6월 1일

정보과학회논문지: 시스템 및 이론 제36권 제5호(2009.10)

1. 서 론

최근 고속 네트워크와 저장 기술의 발전으로 인하여 인터넷 기반 서비스에서 데이터는 점차 대용량화되어 가고 있으며, 대용량 데이터를 효과적으로 처리할 수 있는 플랫폼 기술에 대한 관심이 높아지고 있다. 이미 구글에서는 자사에서 개발한 플랫폼의 주요 속성인 GFS[1], Bigtable[2], MapReduce[3]를 논문으로 발표하여 큰 주목을 받았으며, 구글 플랫폼은 수십 개의 데이터센터에 수십만 대 이상의 PC로 구성된 클러스터 시스템으로 알려져 있다. 이와 함께 Apache에서는 Hadoop이라는 명칭으로 구글 플랫폼 복제화 프로젝트를 진행하고 있다.

이러한 대규모 분산 컴퓨팅 플랫폼은 구글에서 제안한 분산 파일시스템, 분산 데이터베이스, 분산 프로그래밍 모델의 개념을 도입하여 발전하고 있으며, 특히 Hadoop은 오픈 소스를 기반으로 최근 급속하게 발전하고 있다.

본 논문에서는 구글 플랫폼, Hadoop과 같은 대규모 분산 컴퓨팅 환경에서 기존의 TCP를 대체하는 새로운 네트워크 프로토콜의 적용을 제안한다. 수천 대 이상의 노드가 연결된 분산 컴퓨팅 환경에서는 각 노드 사이의 데이터 및 제어신호 전달을 위한 고속 네트워크 프로토콜이 필수적이다. 그러나 기존에 사용되고 있는 TCP는 클러스터 시스템의 내부 통신에 특화되어 있지 않기 때문에, 이를 대체할 수 있는 네트워크 프로토콜을 통해 플랫폼 내부의 통신 성능을 향상시키는 것이 중요한 문제이다. 전통적인 TCP를 개선하는 연구들에는 TCP 혼잡제어 기법을 다양화한 TCP variants, 차세대 전송계층 프로토콜로 고려되고 있는 SCTP, 그리고 클러스터 환경을 위해 개발된 TIPC가 있다. 하지만 이러한 네트워크 프로토콜을 실제 분산 컴퓨팅 환경에서 비교하기 위해서는 몇 가지 제약사항이 존재한다. 구글 플랫폼은 소스코드가 공개되지 않아 현재 활용할 수 없으며, Hadoop은 자바로 구현되어 JDK에서 지원하지 않는 SCTP와 TIPC 프로토콜을 Hadoop에 적용하는 것은 구현상의 복잡도가 매우 높다. 따라서 다양한 프로토콜을 실제 환경에 적용하기에 앞서 이러한 프로토콜 중 분산 컴퓨팅 환경에 가장 적합한 프로토콜이 무엇인지를 찾아내는 것이 중요하다. 본 논문에서는 앞서 언급한 네트워크 프로토콜을 보다 용이하게 적용할 수 있으며 실제 환경이 가진 특성을 갖춘 실험모델을 구현하였다. Hadoop의 주요 속성인 HDFS, MapReduce의 동작방식을 분석하고, 실제로 Hadoop을 설치 및 실행하여 내부에 발생하는 모든 트래픽을 캡쳐하였다. 이를 통해 플랫폼 내부의 네트워크 I/O 발생 구간을 분류하였으며, 이와 유사한 트래픽을 발생시키도록 실험환경을 모델화하였다. 그리고 이 환경에서 TCP variants, SCTP, TIPC를 비교하여 TIPC가 Hadoop 기반 분산 컴퓨팅 환경에 가장 적합한 프로토콜임을 입증하였다.

본 논문은 다음과 같이 구성된다. 2장에서는 관련 연구를 소개하고, 3장에서는 분산 컴퓨팅 환경의 특징을 설명한다. 4장에서는 분산 컴퓨팅 환경을 모델화하기 위한 요소를 설명한다. 5장에서는 실험 및 결과를 서술하고, 마지막 6장에서 결론과 향후 연구를 제시한다.

2. 관련연구

2.1 고속 네트워크를 위한 기술

모든 노드가 네트워크로 연결되어 있는 분산 컴퓨팅 환경에서 고속 네트워크 기술은 매우 중요한 요소이다. 현재 고성능 클러스터 내부의 고속 네트워크 기술에는 InfiniBand, Myrinet, 그리고 Internet Wide Area RDMA Protocol(iWARP)이 주로 쓰이고 있다. 그러나 이러한 기술들은 성능은 뛰어나지만 고가의 독자적인 NIC과 스위치를 필요로 하기 때문에, Hadoop과 같은 대규모의 분산 컴퓨팅 환경에 적용하는 것은 비용 측면에서 문제를 가지고 있다. 이처럼 고가의 네트워크 장비를 이용하여 네트워크를 구성하기 힘든 저비용, 대규모의 분산 컴퓨팅 환경에서는 현재 가격대 성능비가 우수한 Gigabit-Ethernet(GbE)과 TCP/IP 프로토콜이 사용되고 있다.

2.2 Gigabit-Ethernet(GbE) 기반 네트워크 프로토콜

2.2.1 TCP variants

TCP variants[4]는 기존의 TCP를 혼잡제어 기법 측면에서 개선하여 보다 높은 성능을 내도록 하기 위한 프로토콜들이다. 대표적인 예로써 BIC-TCP, CUBIC-TCP가 있으며 주로 장거리, 고 대역폭 네트워크인 LFN (Long Fat-pipe Network) 환경을 대상으로 개발되고 있다. TCP variants는 물리적 대역폭 확장으로 인한 대역폭의 활용률 저하 문제를 해결하기 위하여 혼잡 윈도우 크기를 빠른 속도로 증가시키는 알고리즘을 채택하고 있다. CUBIC-TCP는 현재 전 세계적으로 40% 이상의 인터넷 서버에서 사용 중이며, 리눅스의 기본 혼잡제어 기법으로 사용되고 있다.

2.2.2 SCTP

TCP를 대체할 수 있는 새로운 전송 계층 프로토콜로써 SCTP(Stream Control Transmission Protocol)[5]가 제안되었다. SCTP는 초기에 VoIP 신호전달을 위해 개발되었으며 현재 멀티미디어 웹 응용, 신뢰성을 요구하는 응용서비스 등에 사용되고 있다. 주요 특징으로는 multi-homing, multi-streaming, message-oriented, 그리고 one-to-one과 one-to-many 소켓 API를 지원한다. Multi-homing은 하나의 세션에 다수의 NIC을 통한 다수의 IP 연결을 유지하여 장애에 대비한 대체경로를 확보하는 것이다. 관련 연구[6]에서는 multi-homing의 측면에서 SCTP와 TCP의 성능을 비교하였는데, 패킷 손실률이 높아질수록 하나의 세션에 두 개 이상의 IP 주소를 바인딩하고 있는 SCTP가 TCP에 비해 낮은 전송 지연시간을 가지는 것을 보였다. Multi-streaming은 하나의 세션에 여러 스트림을 전송함으로써 TCP가 가지고 있는 head-of-line blocking 문제를 해결할 수 있다. 관련 연구[7]에서는 손실을 가진 네트워크 환경에서 복수개의 스트림을 유지할 때 보다 높은 전송률을 가진다고 제안한다. 그리고 SCTP는 message-oriented 방식으로 인해 메시지 단위로 통신하는 MPI, RDMA, TIPC 등과 같은 상위 프로토콜의 지원이 용이하며, one-

to-many 소켓 지원은 복수개의 노드와 통신하기 위한 효과적인 소켓 관리 기법을 제공한다.

2.2.3 TIPC

TIPC(Transparent Inter-Process Communication) [8]는 초기 Ericsson사에서 자사의 클러스터를 위한 프로토콜로 개발되었으며, 현재는 오픈 소스화되어 리눅스 커널에 포함되어 있다. TIPC는 loosely coupled 클러스터 시스템에 적합하게 설계되었으며, 단일 노드내의 메시징 기술인 IPC를 다수의 노드간 메시징에 동일하게 적용한 기술이다. 세부적인 특징으로 TIPC는 경량화된 연결설정 과정을 가진다. TCP의 경우 연결 설정 단계에서 최소 9개의 패킷을 주고받는 반면 TIPC는 1개 또는 2개의 패킷만을 주고받음으로써 연결 설정이 이루어진다. 그리고 스택 크기 측면에서 IP 스택보다 작은 footprint의 경량 프로토콜이다. 어드레싱 방법으로는 <Zone, Cluster, Node> 구조의 논리적이고 계층적인 주소체계를 사용한다. 이를 통해 IP 기반 프로토콜의 동적 주소 변환 문제로 인한 부하를 줄일 수 있다. 37대 노드를 이용해서 TIPC가 이식된 MPI 환경과 기존의 TCP 기반 MPI 환경의 성능을 비교한 연구[9]에서는, 연결 설정 과정이 간편하며 동작 방식이 단순한 TIPC를 기반으로 한 환경이 기존의 TCP와 비교해 낮은 지연시간을 가지고 CPU 부하가 적음을 보였다. 그리고 TIPC는 표준 소켓 API를 통한 메시지와 바이트 스트림 기반의 통신 방식 모두를 지원하는 특징을 가진다.

3. 분산 컴퓨팅 환경의 특징

3.1 분산 컴퓨팅 플랫폼, Hadoop

오픈 소스 기반의 Apache Hadoop 프로젝트는 대규모 분산 컴퓨팅 환경의 대표적인 플랫폼이다. 구글 플랫폼의 GFS, Bigtable, MapReduce와 같은 속성을 유사하게 구현하고 있으며, 현재 야후를 중심으로 활발하게 개발 및 활용되고 있다. Hadoop은 크게 HDFS, HBase, MapReduce의 세 가지 주요 속성으로 구성되어 있다. 본 장에서는 이들 중 네트워크 I/O와 실제적으로 관련 있는 분산 파일시스템(HDFS)과 분산 프로그래밍 모델(MapReduce)을 설명한다.

3.1.1 HDFS(Hadoop Distributed File System)

HDFS는 테라 또는 페타 바이트 크기의 대용량 데이터를 저비용이면서 안정적으로 저장할 수 있는 분산 파일시스템이다. HDFS에서 대용량 파일은 일반적으로 64 MB 크기의 블록 단위로 나누어져서 분산 노드에 저장된다. HDFS는 Namenode와 Datanode라 불리는 마스터/슬레이브 구조로 이루어져 있다. 마스터인 Namenode는 파일시스템의 메타데이터를 관리하고, 클라이언트와 Datanode 사이의 데이터 I/O를 제어한다. 그리고 데이터 블록의 손실에 대비하여 Datanode에서의 블록 복사를 제어한다. Namenode를 제외한 다른 모든 노드는 슬레이브 역할의 Datanode이다. Datanode는 실제 데이터를 저장하며, 클라이언트와의 데이터 I/O 및 블록 복사를 직접 수행한다. HDFS에서는 노드의 결함으로 인한 데이터의 손실에 대비하기 위하여 하나의 블록에 대해서 일반적으로 3개의 복사본을 유지한다. 이로 인해 데이터의 신뢰도와 안정성을 확보할 수 있다. 또한 다수의 클라이언트가 동시에 데이터를 요구하는 경우에도 빠르게 데이터를 제공해 줄 수 있는 장점을 가지고 있다. 그리고 HDFS에서 Namenode와 Datanode는 주기적으로 Heart-beat를 주고받는다. 이를 통해 현재 슬레이브 노드의 상태를 마스터 노드가 알 수 있도록 하며, 결함이 발생한 블록에 대한 처리를 수행할 수 있다.

3.1.2 MapReduce

MapReduce는 대용량 데이터가 분산 저장된 병렬 클러스터 환경에서 효과적으로 데이터를 처리할 수 있는 프로그래밍 모델이며 소프트웨어 기반 프레임워크이다. HDFS에 저장된 하나의 블록 단위로 하나의 Map 태스크가 수행되며, 분산 노드에서 각자 수행한 태스크의 결과를 Reduce 태스크에서 병합하여 최종 결과를 생성한다. MapReduce 역시 HDFS와 유사하게 마스터/슬레이브 구조를 가진다. 마스터는 태스크를 할당하고 제어하는 Jobtracker이며, 실제 태스크를 수행하는 슬레이브는 Tasktracker라 불린다. Tasktracker가 수행하는 태스크는 Map과 Reduce 태스크로 나뉘어진다.

그림 1은 MapReduce의 과정을 나타내고 있으며, Tasktracker가 수행하는 Map과 Reduce 태스크간의 관계를 보여준다. 파일은 Split이라고 표시된 블록으로 나누어져 분산 저장되어 있고 각 Tasktracker는 HDFS에서 블록을 읽어 Map 태스크를 수행한다. Map의 결과는 중간 파일 형태로 각각의 노드에 저장되며, Reduce 태스크를 수행하는 노드는 다수의 노드로부터 Map의 결과를 전달받아 MapReduce 최종 결과물을 생성한다.

그림 2는 MapReduce의 동작을 순차적으로 설명하고 있다. (1)에서 하나의 노드에는 3개의 Map 태스크가 할

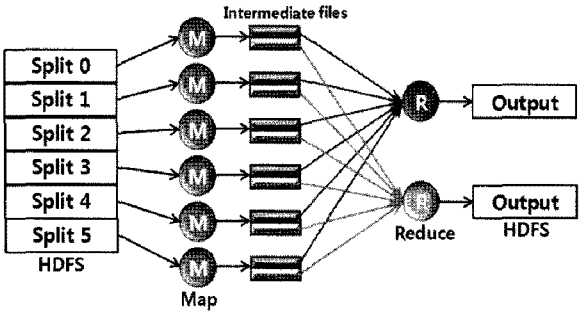


그림 1 MapReduce에서 태스크간의 관계

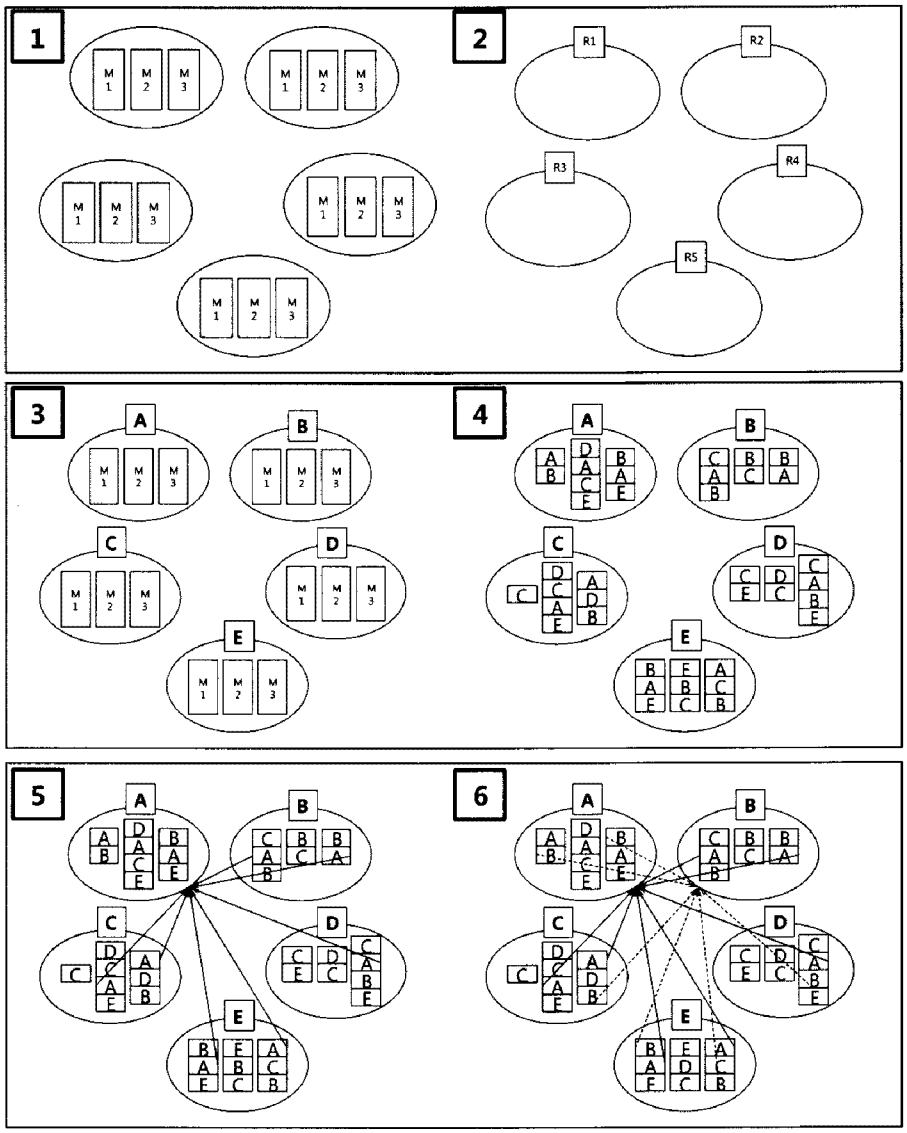


그림 2 MapReduce 수행과정

당되었으며, (2)과 같이 각 노드에서는 1개의 Reduce 태스크를 수행한다. Map 태스크의 결과는 Key와 Value의 쌍으로 이루어지며, Reduce 태스크는 (3)의 A~E와 같이 자신이 병합하게 될 Key를 선정한다. (4)에서 각 Map 태스크의 결과로 A~E의 Key를 가진 데이터가 생성되었으며, 이들 데이터는 (5), (6)과 같이 각각의 Key를 병합하는 Reduce 태스크로 전달되게 된다.

3.2 MapReduce 과정에서 발생하는 네트워크 I/O

본 논문에서는 Hadoop에서 발생하는 네트워크 I/O의 부하를 분석하기 위해 네트워크 I/O 발생 구간을 분류하고, 실제 Hadoop을 설치 및 실행하여 내부에 발생하는 트래픽을 측정하였다. Hadoop의 설치 환경은 다음과 같다. 마스터 노드 1대, 슬레이브 노드 7대의 총 8대 노드에 Hadoop-0.18.0-release와 JDK 1.6 버전을 설치하였으며, 블록 복사 횟수 3회, 블록 크기 64 MB, 그리고 Heartbeat 주기는 3초로 설정하였다. Hadoop의 실행을 위해서 약 2.5 Gigabyte 크기의 텍스트 파일을 임의로 생성하였다. 그리고 파일을 HDFS에 저장하여 파일내의 단어 개수를 세는 Wordcount[10]를 수행하였고, tcpdump [11]를 이용해 이때 발생하는 모든 트래픽을 캡쳐하였다.

그림 3은 MapReduce 과정에서 데이터 전달을 위해 발생하는 네트워크 I/O 구간을 보여준다. 네트워크 I/O는 그림에서와 같이 크게 2개의 구간에서 발생한다. ①은 Tasktracker에서 Map 태스크를 수행하기 위해 HDFS의 블록을 읽는 구간이며, ②는 Tasktracker 사이에서 Map의 결과를 Reduce 태스크에게로 전달한다.

3.2.1 Map을 위한 HDFS읽기

Tasktracker는 HDFS로부터 해당하는 데이터 블록을 읽어와 Map 태스크를 수행한다. Hadoop에서는 기본적으로 자신의 로컬 블록에 대한 태스크를 할당 받기 때

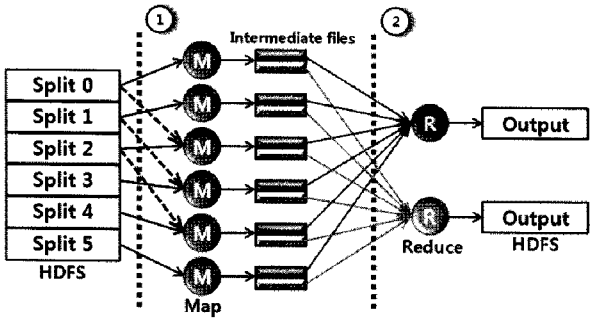


그림 3 Hadoop내의 네트워크 I/O 구간

문에 먼저 로컬 디스크에서 블록을 조회하고 읽어온다. 그러나 상황에 따라서 그림 3의 ①과 같이 Map 태스크를 수행하기 위해 원격지 노드의 블록을 읽어 오는 경우가 발생한다. 이러한 상황은 전체 플랫폼의 노드 중 일부만 MapReduce에 활용되는 경우, 노드 간 성능차이가 있는 경우, 그리고 데이터 블록이 모든 노드에 고르게 분산되어 있지 않은 경우 등이 있다.

그림 4는 단순하게 구성한 실제 Hadoop 환경에서 Wordcount 수행 시 임의로 선정된 Map 태스크의 TX 트래픽이다. 실험 경과시간에 따라 일정한 주기로 TX 트래픽이 발생하는데 이는 Map의 결과를 Reduce 태스크에게 전송하는 것이다. 실험 경과시간 후반부에서는 burst한 TX 트래픽이 발생한다. 이 부분이 원격지 노드로 데이터 블록을 전달하는 것이며, Tasktracker 간에 1:1 데이터 통신이 발생함을 알 수 있다.

3.2.2 Map 결과를 Reduce 태스크로 전달(Shuffle)

Reduce 태스크를 수행하는 Reducer는 Map태스크를 수행하는 Mapper로부터 수행이 완료된 Map의 결과를 전달받는 Shuffle을 수행한다. 그림 3의 ②와 같이 Shuffle 과정에서 Reducer의 요구를 받은 Mapper들은 중간 파일로 저장된 Map의 결과를 Reducer에게 송신한다. Reducer 내에는 다수의 Mapper들과 연결을 맺고 데이터를 요구하는 다수의 thread가 존재하며, 이로 인해 데이터는 Reducer에게 병렬적으로 전달된다. Hadoop Cluster Setup 문서[12]에 따르면 900대와 2000대 규모의 Hadoop 플랫폼에서 Mapper로부터 Reducer가 수신하는 병렬적인 데이터 전송 수를 20, 50개로 설정하여 벤치마크를 수행하였다. 대규모 분산 컴퓨팅 환경에서는 이와 같이 수십 대의 노드가 하나의 노드로 데이터를 동시에 전송하는 네트워크 트래픽이 발생한다. 그리고 Shuffle에서 Reducer와 Mapper간 요구/응답 관계의 통신을 위한 상위 프로토콜로 HTTP가 사용된다. Hadoop의 Tasktracker 내에 Jetty 웹 서버가 내장되어 Shuffle 과정의 HTTP 통신을 지원한다. HTTP는 Shuffle의 통신에서 발생하는 요구/응답 모델에 적합하게 설계되었으며, Persistent Connection의 특성[13]으로 인해 일정시간 동안 연결이 유지되는 장점을 가지고 있다.

그림 5는 Wordcount 수행 시 Reducer의 RX 그래프이다. 이 그래프를 통해 Shuffle 과정에서의 Mapper와 Reducer 간 병렬적인 데이터 전송을 볼 수 있다. 이 측정에서는 총 8대 노드의 소규모 실험환경하에서 총 8개의 태스크만을 설정하여Hadoop을 실행하였다. Mapper로부터 Reducer가 수신하는 병렬적인 전송 수(mapred.reduce.parallel.copies)는 최대 5로 설정하였으며, 이로 인해 병렬적 데이터 수신 시 Reducer의 네트워크 I/O 부하가 크지 않았다. 하지만 이러한 환경이 대규모로 구

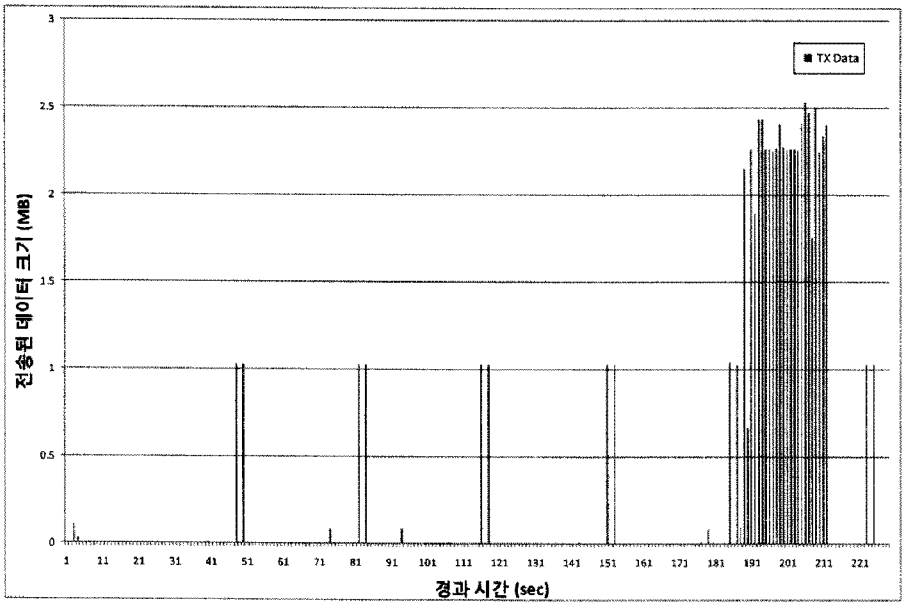


그림 4 Map 수행 시 측정한 TX 데이터

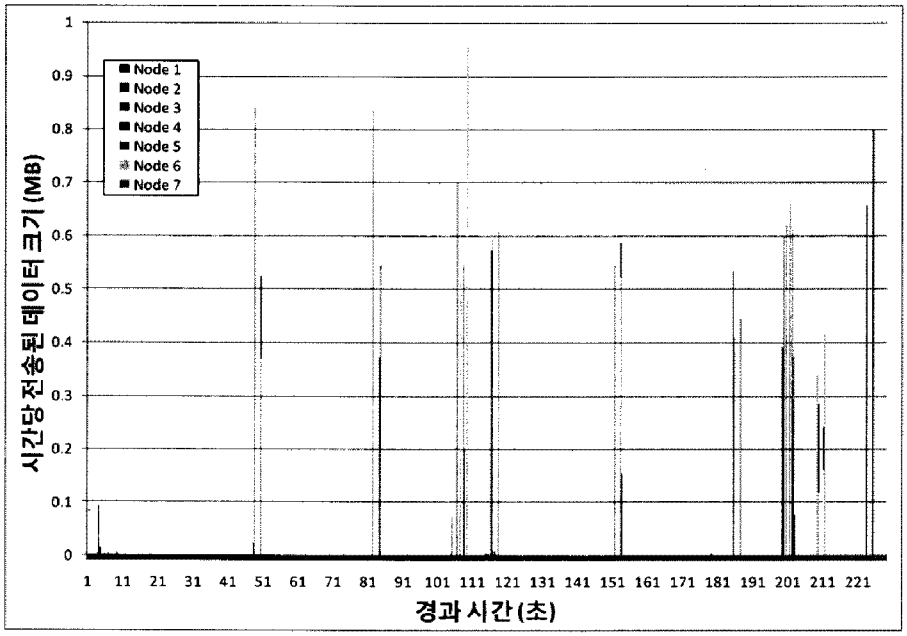


그림 5 Reduce 태스크 수행 시 RX 데이터

축되면 동시에 수십 대의 노드에서 하나의 노드로 트래픽이 집중되며 보다 빈번한 트래픽이 발생하게 되기 때문에 이 부분에서의 네트워크 부하를 고려하는 것이 중요하다.

3.3 MapReduce 태스크 규모 및 분포

Hadoop MapReduce Tutorial[14]에서 제안한 바에 따르면 일반적인 Map 태스크 개수는 하나의 노드 당 동시에 10~100개의 태스크가 수행되는 것이다. 그리고 CPU 사용률이 매우 낮은 태스크에 한해서는 최대 300개의 Map 태스크를 동시에 수행하도록 제안하고 있다. 이러한 이유는 Map 태스크가 초기 setup 시간이 길기 때문에 순차적으로 수행하는 것보다 병렬화 수준을 높이는 것이 효율적이기 때문이다. 그리고 Reduce 태스크의 경우 전체 플랫폼에서 수행되는 Reduce 태스크의 개수는 간단한 두 개의 식으로 나타내어지고 있다.

① 0.95 x 노드 수 x 하나의 노드에서 동시에 수행되는 Reduce 수

② 1.75 x 노드 수 x 하나의 노드에서 동시에 수행되는 Reduce 수

식의 요소 중 하나의 노드에서 동시에 수행되는 Reduce 개수는 특별히 정해진 바 없지만, 하나의 노드에서 한 개 이상의 Reduce가 수행될 때 전체 플랫폼에 수행되는 Reduce 수는 전체 노드 수의 0.95~1.75배 이상으로 볼 수 있다. 그리고 구글에서 발표한 논문[15]에 따르면 Reduce태스크의 개수는 전체 노드 수의 small multiple개라고 정의하고 있어, Reduce의 경우 Map보다 적은 수의 태스크가 수행됨을 알 수 있다.

이러한 MapReduce구조에서는 Map과 Reduce를 수행하는 노드가 분리되지 않고 하나의 노드에서 Map과 Reduce가 동시에 수행된다. 다수의 Map 과 Reduce의 Shuffle을 동시에 수행하는 것은 Map 태스크의 처리시간에 영향을 미칠 수 있다. 그 이유는 Shuffle에서의 네트워크 I/O를 처리하기 위해 CPU 사용률이 증가하고, 이로 인해 Map 태스크의 처리시간이 증가하기 때문이다. 따라서 MapReduce 수행 시 네트워크 I/O의 부하 정도가 전체 플랫폼의 성능에 미치는 영향이 크기 때문에 네트워크 I/O 부하를 경감하는 것은 매우 중요한 요소라고 볼 수 있다.

4. 분산 컴퓨팅 환경의 모델화를 위한 고려사항

본 논문에서는 분산 컴퓨팅 환경에서 네트워크 프로토콜의 성능을 비교하기 위해 Hadoop에 기반한 실험모델을 구현하고 이 환경에서 몇 가지 프로토콜을 적용 및 실험하였다. 앞 장에서 다룬 바와 같이 Hadoop의 네트워크 I/O 구간을 분류 및 분석하였고 그 중 가장 통신 부하가 큰 Reducer의 Shuffle 과정을 중심으로 실험모델을 설계하였다. 실험모델의 구현에서 고려해야 할 사항은 크게 3가지로 나누어진다. 첫 번째는 1:N 관계의 network-intensive 트래픽의 발생이며, 두 번째는 네트워크 I/O의 발생 주기이고, 마지막으로 실제적인 환경에서 플랫폼 내부에 발생하는 데이터 크기를 고려해야 한다.

4.1 1:N 통신

분산 컴퓨팅 환경을 모델화한 실험환경의 첫 번째 특징은 1:N 관계의 network-intensive 트래픽을 모델화한 것이다. Reducer 측에서는 N개의 Mapper와 연결을 맺

고 통신을 수행할 수 있도록 N개의 프로세스를 *fork()* 한다. 그리고 각 Mapper와 연결을 맺은 Shuffle 프로세스들은 동시에 데이터 전송을 요구하고 수신모드로 전환하여 데이터 수신을 기다린다. 그림 6은 1:N 관계의 통신에서 다수의 Mapper에게 데이터를 요구하고 그 이후 동시에 데이터를 수신하는 과정을 나타내고 있다.

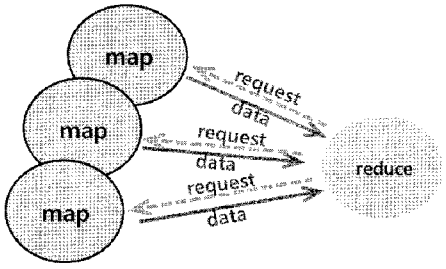


그림 6 1:N 통신모델

4.2 MapReduce 수행 시 트래픽 발생주기

MapReduce에서 발생하는 Network I/O의 주기는 노드 수와 동시에 수행되는 태스크 개수에 의존적이다. 3.3장에서 다루었던 MapReduce 태스크 규모 및 분포에 따르면 하나의 노드에 10~100개의 Map 태스크와 1개 이상의 Reduce 태스크가 수행되며 이러한 특성을 기반으로 트래픽 발생 주기를 예측해 볼 수 있다. 표 1은 구글에서 발표한 논문[15]에서 제시하는 MapReduce 수행 규모를 나타내고 있다. 예를 들어, 표 1과 같은 환경에서 Map 태스크의 평균 수행시간을 100초, 동시에 Reducer가 수신하는 병렬 전송 수를 50으로 가정하면 약 25ms 간격으로 1:N 통신이 발생됨을 알 수 있다. 본 논문에서는 데이터 전송 주기와 병렬 전송 수를 조절하여 각 네트워크 프로토콜의 성능을 측정할 수 있도록 실험모델을 설계하였다.

표 1 MapReduce 수행 규모

노드 개수	2,000
블록 크기 (MB)	16~64
Map 태스크 수	200,000
Reduce 태스크 수	5,000

4.3 플랫폼 내부에 전송되는 데이터 크기

하나의 Map output은 Key에 따라 여러 개의 파티션으로 나뉘어 다수의 Reducer로 전달된다. Map output을 Key에 따라 파티션으로 분할하는 작업을 Partitioner가 담당하고 있으며 Hadoop에서 디폴트 파티션 기법은 Hash를 사용하고 있다. 각 Reducer는 독립적인 Key를 갖고 있으므로 Key의 개수는 전체 플랫폼에 존재하는 Reduce 태스크의 수가 된다. 따라서 Map output은 전체 플랫폼에 존재하는 Reduce 태스크 개수만큼 쪼개어지게 되고 Shuffle 수행 시 각각의 파티션이 하나의 Reducer로 전달된다. 그러므로 2000개의 Reduce 태스크가 존재한다면 하나의 파티션 조각 크기는 평균적으로 Map output 크기의 1/2000 정도 될 수 있다.

평균 Map output 크기는 Map input 크기와 관련되어 있다. 표 2는 구글에서 발표한 MapReduce Statistics[15]로 Map input, output, 그리고 Reduce output 크기에 대한 통계를 나타낸다. 구글 통계에 따르면 Map output 크기는 Map input의 10% 정도인 것으로 분석해 볼 수 있다. 여기서 Map input 데이터의 크기를 64MB로 설정할 경우 평균적인 Map output의 크기는 약 6.4MB가 되며 2000개의 Reduce 태스크가 존재한다면 Shuffle 수행 시 전달되는 하나의 파티션 조각 크기는 평균적으로 약 3~4KB가 됨을 알 수 있다. 본 논문에서는 Shuffle 과정에서 발생할 수 있는 다양한 데이터 크기에 대해 실험할 수 있도록 실험환경을 설계하였다.

표 2 구글 MapReduce Statistics

	Aug. '04	Mar. '06	Sep. 07
Map input data (TB)	3,288	52,254	403,152
Map output data (TB)	758	6,743	34,774
Reduce output data (TB)	193	2,970	14,018

5. 실험 및 결과

본 논문에서는 대규모 분산 컴퓨팅 환경에서 다양한 네트워크 프로토콜을 비교실험하기 위해 Hadoop을 분석하여 그와 유사한 실험환경을 모델화하였다. 그리고 관련연구에서 소개한 GbE 기반 네트워크 프로토콜을 실험모델에 적용하여 평균 전송률과 CPU 점유율을 비교하였다. 실험에 사용된 시스템 환경은 다음과 같다. AMD Opteron 1210 51대에 2.6.25 커널 버전의 리눅스를 설치하였으며, 각 실험 노드는 Tigon3 GbE 컨트롤러를 장착하고 기가비트 네트워크 연결을 위해 3Com® Baseline Switch 2848-SFP 스위치를 사용하였다. 실험에 쓰인 프로토콜은 CUBIC-TCP, TIPC 1.7.6, Linux Kernel SCTP 1.0.9이며 전송률 계산에 필요한 시간 측정은 *gettimeofday()* 함수, CPU 점유율 측정에는 *sar* 커맨드를 이용하였다.

그림 7은 Reducer의 평균 전송률을 비교한 그래프이다. 이 실험에서는 병렬전송을 발생시키는 노드 수와 데이터 크기를 변화시키며 각각의 경우에 1000회의 반복실험을 수행하였다. 그리고 Reducer는 동시에 모든 Mapper에게 데이터 요구 메시지를 전송하고, 그때부터 마지막 데이터를 수신할 때까지의 시간을 측정하여 평균 전송률을 계산하였다. 실험 결과 대부분의 경우에서

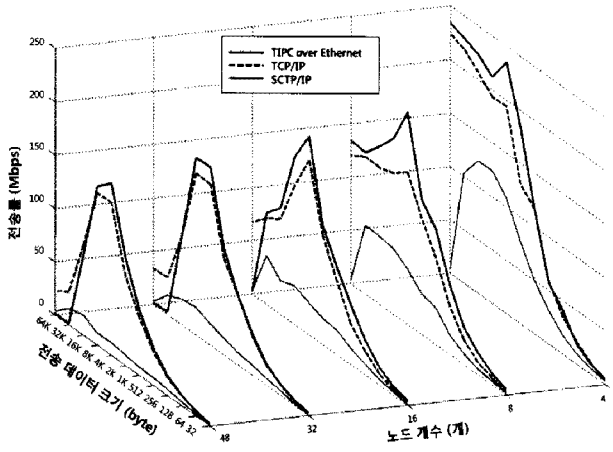


그림 7 노드 수와 데이터크기에 따른 평균 전송률

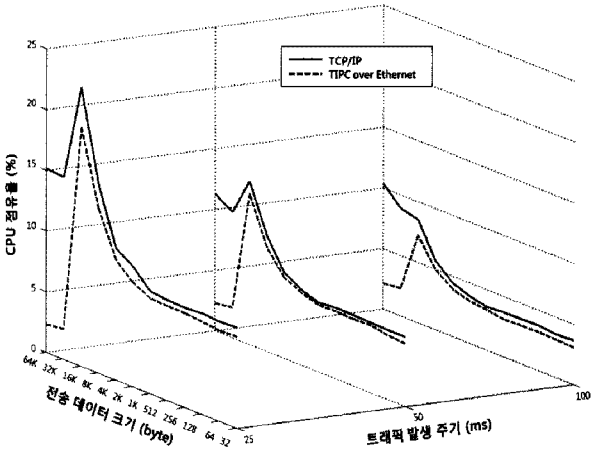


그림 8 트래픽 주기와 데이터크기에 따른 CPU 점유율

TCP, SCTP와 비교해 TIPC가 우수한 성능을 보였으나, 병렬전송 수가 증가하고 데이터크기가 큰 경우에 TIPC는 낮은 성능을 보였다. 병렬전송 노드가 48개인 1:48 통신의 경우 8KB 이하 구간에서 TIPC는 TCP보다 평균 10%, 최대 16%의 높은 전송률을 가졌으나, 네트워크 혼잡도가 높아지는 16KB 이상의 구간에서는 TIPC의 성능이 매우 낮았다. 이러한 원인은 TIPC가 혼잡윈도우를 관리하는 부하를 줄이기 위해 정적 혼잡윈도우 기법을 사용[9]하고 있기 때문으로, TIPC는 네트워크 혼잡이 심해질 경우에 성능이 급격히 저하되는 단점을 가지게 된다. 그러나 본 논문의 4.3장에서 분석한 바와 같이 수십 개의 병렬전송이 발생하는 수천 대 규모의 분산 컴퓨팅 플랫폼에서는 MapReduce 수행 시 Map output이 매우 작은 크기의 파티션으로 분할되기 때문에 8KB 이하의 구간에서 높은 성능을 내는 것이 더욱 중요한 요소라고 볼 수 있다.

그림 8은 1:50 통신 환경에서 통신부하 발생 주기에 따른 CPU 점유율을 측정한 그래프이다. 트래픽이 빈번하게 발생할수록 CPU 점유율이 높았으며, 4.2장에서 언급한 2,000대 노드의 MapReduce에서 발생하는 25ms 주기의 경우 16KB 이하 크기에서 TIPC는 TCP보다 7~15%의 낮은 CPU 점유율을 보였고, 그 이상의 구간에서는 TIPC의 낮은 전송률로 인해 CPU 점유율도 급격히 낮아짐을 알 수 있었다.

6. 결 론

본 논문에서는 대규모 PC 클러스터상의 분산 컴퓨팅 환경을 위한 고속 네트워크 프로토콜로 TCP/IP를 대신하여 TIPC의 적용을 제안한다. 이를 뒷받침하기 위해 분산 컴퓨팅 환경의 대표적 플랫폼인 Hadoop을 분석하여 분산 컴퓨팅 환경을 모델화하였으며, 이 실험환경에서 CUBIC-TCP, SCTP, 그리고 TIPC 등 여러 프로토콜을 비교실험 하였다. 실험 결과 평균 전송률 측면에서 TCP, SCTP와 비교해 TIPC의 성능이 가장 우수하였으며, TIPC는 TCP보다 낮은 CPU 점유율을 보여 TIPC 적용 시 네트워크 I/O의 부하 감소로 인한 태스크 처리 시간의 단축을 예상해 볼 수 있다.

향후 과제로는 네트워크 혼잡 상황에서 보다 나은 대처를 위한 TIPC의 혼잡 윈도우 관리 기법의 개선을 들 수 있다. 그리고 TIPC를 실제 Hadoop에 적용할 수 있도록 Java TIPC Library를 개발하여 실제 환경에서 성능을 비교해 보는 것이 본 연구의 남겨진 과제이다.

참 고 문 헌

[1] S. Ghemawat, H. Gobioff, and S.-T. Leung, "The Google file system," 19th ACM Symposium on Operating Systems Principles, Oct. 2003.

[2] F. Chang, J. Dean, S. Ghemawat, W. C. Hsieh, D. A. Wallach, M. Burrows, T. Chandra, A. Fikes, and R. E. Gruber, "BigTable: A distributed storage system for structured data," *Seventh Symposium on Operating System Design and Implementation*, pp.725-726, 2006.

[3] J. Dean and S. Ghemawat, "Mapreduce: Simplified data processing on large clusters," *Sixth Symposium on Operating System Design and Implementation*, pp.137-150, 2004.

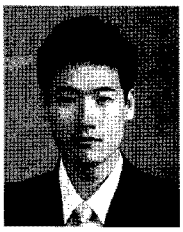
[4] S. Ha, L. Le, I. Rhee, and L. Xu, "Impact of background traffic on performance of high-speed tcp variant protocols," *Computer Networks*, vol.51, no.7, pp.1748-1762, 2007.

[5] R. Stewart and C. Metz, "SCTP: New transport protocol for TCP/IP," *IEEE Internet Computing*, vol.5, no.6, pp.64-69, November/December. 2001.

[6] 송정화, 이미정, 고석주, "SCTP의 멀티호밍 특성에 대한 성능 평가", *한국정보처리학회논문지*, 제11-C권 제2호, pp.245-252, 2004년 4월.

[7] 박재성, 고석주, "리눅스 환경에서 SCTP와 TCP 프로

토콜의 성능 비교", *한국통신학회논문지*, 제33권 제8호, pp.699-706, 2008년 8월.
[8] J. P. Maloy, "TIPC: Providing Communication for Linux Clusters," Linux Symposium, vol.2, pp.347-356, 2004.
[9] Stylianos Bounanos, Martin Fleury, "Gb Ethernet Protocols for Clusters: An OpenMPI, TIPC, GAMMA Case Study," The Parallel Computing Conference (ParCo), pp.397-404, 2007.
[10] Hadoop Wordcount Example, Available: http://wiki.apache.org/hadoop/WordCount
[11] Van Jacobson and etc, The protocol packet capture and dumper program, Available: http://www-nrg.ee.lbl.gov/nrg.html
[12] Hadoop Cluster Setup Document, Available: http://hadoop.apache.org/core/docs/r0.18.0/cluster_setup.html
[13] RFC 2616, part of Hypertext Transfer Protocol-HTTP/1.1
[14] Hadoop MapReduce Tutorial, Available: http://hadoop.apache.org/core/docs/r0.18.0/mapred_tutorial.html
[15] Jeffrey Dean and Sanjay Ghemawat, "MapReduce: Simplified Data Processing on Large Clusters," COMMUNICATIONS OF THE ACM, vol.51, no.1, pp.107-113, January 2008.



유 대 현
2007년 부산대학교 컴퓨터공학과 학사
2009년 부산대학교 컴퓨터공학과 석사
2009년~현재 삼성탈레스 SW팀 연구원
관심분야는 클러스터 시스템, TOE, Hadoop

정 상 화
1985년 서울대학교 전기공학과 학사. 1988년 Iowa State Univ. 컴퓨터공학과 석사 1993년 Univ. of Southern California 컴퓨터공학과 박사. 1993년~1994년 Univ. of Central Florida 컴퓨터공학과 조교수. 1994년~현재 부산대학교 컴퓨터공학과 교수, 컴퓨터 및 정보통신연구소 연구원 2002년~2003년 Oregon State Univ. 컴퓨터공학과 초빙교수. 관심분야는 클러스터 시스템, TOE, RDMA, WMN, RFID

김 태 훈
2008년 2월 부산대학교 컴퓨터공학과 학사. 2008년 3월~현재 부산대학교 컴퓨터공학과 석사과정. 관심분야는 네트워크, WMN, 이동통신망