http://dx.doi.org/10.5392/JKCA.2016.16.02.163

빅데이터 환경에서 스트림 질의 처리를 위한 인메모리 기반 점진적 처리 기법

In-Memory Based Incremental Processing Method for Stream Query Processing in Big Data Environments

복경수, 육미선, 노연우, 한지은, 김연우, 임종태, 유재수
충북대학교 정보통신공학과

Kyoungsoo Bok(ksbok@chungbuk.ac.kr), Misun Yook(misun@chunhbuk.ac.kr),
Yeonwoo Noh(ywnoh@chungbuk.ac.kr), Jieun Han(jieun24@chunhbuk.ac.kr),
Yeonwoo Kim(sage106@nate.com), Jongtae Lim(jtlim@chungbuk.ac.kr),
Jaesoo Yoo(yjs@chungbuk.ac.kr)

요약

최근 대용량의 스트림 데이터를 분산 처리하기 위한 연구들이 진행되고 있다. 본 논문에서는 빅데이터 환경에서 실시간 스트림 데이터의 점진적 처리 기법을 제안한다. 제안하는 기법은 처음 스트림 데이터가 입력되면 임시 큐에 데이터를 저장하고 마스터 노드에 저장되어 데이터와 비교과정을 통해 마스터 노드에 동일한 데이터가 있는 경우 마스터 노드에서 가지고 있는 노드의 정보를 이용하여 해당 노드의 메모리에서 기존 처리 결과를 재사용한다. 기존 처리 결과가 없다면 처리하고 처리 결과를 메모리에 저장한다. 분산 환경에서 점진적인 스트리밍 데이터 처리를 위해 노드의 작업 지연을 계산하여 노드의 부하를 파악하고 처리 시간 계산을 통해 각 노드의 성능을 고려한 잡 스케쥴링 기법을 제안한다. 제안하는 기법의 우수성을 보이기 위해 기존 기법과의 질의 수행 시간 비교를 위한 성능평가를 수행한다.

■ 중심어 : | 빅데이터 | 인메모리 | 분산 처리 | 실시간 처리 | 스트리밍 데이터 |

Abstract

Recently, massive amounts of stream data have been studied for distributed processing. In this paper, we propose an incremental stream data processing method based on in-memory in big data environments. The proposed method stores input data in a temporary queue and compare them with data in a master node. If the data is in the master node, the proposed method reuses the previous processing results located in the node chosen by the master node. If there are no previous results of data in the node, the proposed method processes the data and stores the result in a separate node. We also propose a job scheduling technique considering the load and performance of a node. In order to show the superiority of the proposed method, we compare it with the existing method in terms of query processing time. Our experimental results show that our method outperforms the existing method in terms of query processing time.

■ keyword : | Big Data | In-memory | Distribute Processing | Real-time Processing | Streaming Data |

* 본 연구는 미래창조과학부 및 정보통신기술진흥센터의 대학ICT연구센터육성 지원사업(IITP-2015-H8501-15-1013), 미래창조과학부 및 정보통신기술진흥센터의 정보통신·방송 연구개발 사업[B0101-15-0266, (딥뷰-1세부) 실시간 대규모 영상 데이터 이해·예측을 위한 고성능 비주얼 디스커버리 플랫폼 개발], 교육부와 한국연구재단의 지역혁신인력양성사업으로 수행된 연구결과임 (No.2013H1B8A2032298), 2013년도 정부(미래창조과학부)의 재원으로 한국연구재단의 지원 (No.2013R1A2A2A01015710)을 받아 수행된 연구임

접수일자 : 2015년 11월 15일
수정일자 : 2015년 12월 08일
심사완료일 : 2015년 12월 08일
교신저자 : 유재수, e-mail : yjs@chungbuk.ac.kr

I. 서 론

IT 기술의 발전과 함께 다양한 정보 채널의 등장으로 생성 및 유통되는 정보량이 기하급수적으로 증가하고 있다. 이와 함께 기존 정보 처리 기술을 통해 데이터 저장, 관리, 분석이 어려워짐에 따라 빅데이터의 중요성이 증가되고 있다[8]. 빅데이터는 거대한 데이터 양(volume), 빠른 데이터 유통 및 이용 속도(velocity), 데이터 다양성(variety)을 특징으로 하며 이러한 빅데이터를 활용하여 새로운 가치 창출을 하기 위해 다양한 분석 및 처리 기법들에 대한 연구가 진행되고 있다[7]. 최근 소셜 네트워크, 위치기반 서비스, M2M 기술에 대한 활용이 증가되면서 대용량의 스트림 데이터 처리에 대한 관심이 집중되고 있다. 스트림 데이터는 각종 장비, 응용 서비스 등에서 지속적으로 데이터가 발생되기 때문에 실시간 데이터 수집 및 처리 방법이 요구된다[8]. 예를 들어, 사건, 사고, 장애 및 시설물 관리 등 다양한 분야에서 방대한 양의 데이터들이 생성되고 있으며 이렇게 생성된 데이터는 모니터링과 같은 실시간 시스템에 이용되고 있다.

빅데이터를 처리하기 위한 대표적인 플랫폼으로는 하둡(Hadoop)[1]이 있다. 하둡[1]은 대용량의 데이터를 분산 처리하고 관리하기 위해 사용되며 대용량 데이터의 저장소인 하둡파일시스템(HDFS)과 데이터 처리하기 위한 맵리듀스(Mapreduce)로 구성된다. 대용량의 데이터를 분산 처리하기 위한 맵리듀스는 데이터를 여러 조각으로 나누어서 처리하는 단계인 맵(Map) 단계와 처리 결과를 모아서 결과를 출력하는 리듀스(Reduce)단계로 나누어져 처리된다. 하둡은 대규모의 데이터를 처리하고 저장하는데 효과적이지만 일괄 배치 처리 시스템이기 때문에 실시간 스트림 데이터를 처리하는데 어려움이 있어 스트림 데이터 처리를 위한 새로운 기법이 필요하다.

실시간으로 입력되는 스트림 데이터를 분석하여 현재 어떤 일이 발생하고 있는지 지속적으로 분석하여 확인하고 대응할 수 있도록 실시간 분석 및 처리가 요구되며 이를 위해 실시간 스트림 데이터 처리에 대한 다양한 연구가 진행되고 있다[2-5][10][11]. 스트림 데이터 처리의 경우 슬라이딩 윈도우를 이용한 처리를 하며 타임 윈도우가 겹쳐서 처리되는 질의 타입의 경우 기존 슬라이딩 윈도우의 처리에서는 중복으로 데이터를 처리하기 때문에 성능이 저하된다. 따라서 과거에 처리된 데이터를 재사용하여 중복 처리를 피하는 효율적인 질의 처리 기법이 필요하다. 분산 환경에서는 노드의 부하와 성능이 고려된 스케쥴링 기법이 필요하며 기존 기법에서는 재사용 데이터를 고려하지 않아 점진적인 처리에 맞는 스케쥴링 기법이 필요하다[6].

스톰[3]은 스케쥴링 기법으로 라운드로빈 스케쥴러를 사용하고 있으며 스파크[2]에서는 다양한 스케쥴러를 사용하고 있으며 대표적으로 FIFO 스케쥴러를 사용한다. 이러한 스케쥴링 기법은 실시간 노드 부하와 기존에 처리된 메모리 데이터를 고려하지 않아 실시간으로 처리하기 어렵다는 단점을 가지고 있다. MapReusing 기법[4]와 Slider기법[5]에서는 점진적인 처리를 위한 연구를 수행하였다. 하지만 분산 환경에서 노드의 실시간 부하 및 성능을 고려하지 않아 성능 향상에 어려움이 있다. 따라서 스트림 데이터의 점진적 질의 처리기법이 요구되며 분산 환경에서 효과적인 점진적인 질의 처리를 위한 실시간 노드 부하와 성능을 고려한 스케쥴링 기법이 요구된다.

본 논문에서는 실시간으로 입력되는 스트림 데이터에 대한 점진적 처리를 지원하는 분산 인메모리 데이터 처리 기법을 제안한다. 제안하는 기법은 스트림 데이터의 점진적 처리를 위해 스트림 데이터가 입력되면 해시키(Hash key) 값을 생성하며 <키, 값> 형태로 임시 큐에 저장한다. 마스터 노드에는 과거 처리 결과의 정보와 해시 키 값이 저장되어 있으며 임시 큐에 있는 해시 키 값과 마스터 노드에 있는 해시 키 값의 정보가 같다면 해당 노드에 접근하여 기존 처리 결과를 재사용한다. 해시 키 값 비교를 통해 동일한 데이터의 경우 메모리에서 처리 결과를 가져와 재사용하며 처리 결과가 저장되어있지 않은 경우 데이터를 노드에 할당하여 처리하고 데이터를 메모리에 저장한다. 기존에 처리된 데이터를 저장하고 재사용하여 데이터 처리 시간을 감소시킨다. 노드의 실시간 부하와 성능을 고려한 스케쥴링 기법을 사용한다. 재사용 가능한 노드를 확인하고 재사

용가능한 노드에 우선 데이터를 할당한다. 다음으로 작업 지연을 계산하여 작업이 많은 노드를 피하고 데이터 처리 시간 계산을 통한 노드의 성능을 고려하여 데이터를 할당한다. 데이터의 점진적인 질의 처리와 실시간 부하 및 성능을 고려한 스케쥴링 기법으로 기존 연구보다 데이터의 처리 시간을 감소시킨다.

본 논문의 구성은 다음과 같다. Ⅱ장에서는 본 논문과 관련된 기존 연구들을 분석하고 Ⅲ장에서는 제안하는 스트림 데이터 처리 기법에 대해 소개한다. Ⅳ장에서는 제안하는 기법의 우수성을 확인하기 위해 성능 평가를 수행한 결과를 기술한다. 마지막 Ⅴ장에서는 논문의 결론 및 향후 연구에 대해 기술한다.

Ⅱ. 관련 연구

하둡[1]은 대용량의 데이터를 분산 처리하고 관리하기 위하여 사용되며 데이터를 분산 저장 및 관리를 위한 하둡파일시스템(HDFS)과 데이터를 처리하기 위한 맵리듀스(Mapreduce)로 구성된다. 하둡 맵리듀스는 빅데이터 처리에서 중요한 플랫폼이지만 문제점을 가지고 있다. 디스크 환경에서 처리되기 때문에 디스크 I/O 및 네트워크 트래픽이 발생하며 이로 인해 중간 데이터 처리 과정에서 심각한 병목 현상이 발생한다. 또한, 일괄 배치 처리를 위해 제안되었기 때문에 실시간 데이터 처리를 통한 분석 결과를 얻는데 한계가 있다.

스톰[2]은 실시간 스트림 방식의 처리 시스템으로 인메모리 컴퓨팅에 초점을 맞추고 실시간 이벤트를 처리한다. 실시간 이벤트 기반 고속 처리를 목적으로 쿼리보다는 처리 중심이다. 스톰[2]은 데이터의 실시간 처리를 위해 스파우트 부분과 볼트 부분으로 구성된다. 스파우트는 스트림 데이터를 입력받는 부분이며 이렇게 입력 받은 데이터를 볼트로 전달하는 역할을 맡고 있다. 볼트는 스파우트로 부터 데이터를 입력받아 처리하며 그 결과를 다른 볼트에게 전달할 수 있어 볼트는 다양한 노드로 데이터를 입력받거나 입력할 수 있다.

스파크[3]는 하둡에서 문제가 되고 있는 디스크 I/O 최소화 하여 병목현상을 해결하는 인메모리 기반 분산 데이터 처리 시스템이다. 배칭 프로세싱 기반 병렬처리 시스템으로 데이터베이스의 운영 및 분석을 위해 데이터의 처리 후 분석을 목적으로 하고 있다. 스파크[3]에서는 RDD라는 개념을 도입하고 있으며 RDD 단위로 데이터를 처리한다. 스톰[2]와 스파크[3]는 인메모리를 기반으로 스트림 데이터를 처리하는데 효과적이지만 점진적인 처리를 지원하지 않는다.

하둡 맵리듀스 환경에서 점진적인 처리를 위한 많은 연구가 수행되었다. MapReusing 기법[4]은 기존에 처리된 데이터를 메모리에 저장하고 재사용하여 데이터 처리 속도를 향상시켰다. 처음 데이터가 입력되면 데이터를 처리하고 결과를 저장한다. 데이터가 입력되면 데이터 비교 과정을 통해 같은 이전에 처리했던 데이터와 같은 데이터인 경우 저장된 데이터를 불러와 사용하고 나머지 저장되지 않은 데이터만 처리한다. 하지만 이에 맞는 데이터 잡 스케쥴링 기법이 적용되지 않아 데이터를 처리하는데 효율적이지 못하다는 단점을 가지고 있다.

실시간 스트림 데이터가 등장하면서 실시간 처리를 위한 스케쥴링 기법들이 연구되었다. [6]은 맵리듀스 환경에서 노드의 부하를 고려한 스케쥴링 기법을 제안하였다. 맵 단계에서 부하가 적은 맵에 데이터를 더 많이 할당하여 데이터를 처리한다. [6]에서는 실시간 부하를 고려하여 효과적인 처리를 제공하지만 기존 데이터를 재사용하는 점진적 처리에는 적합하지 않다. 따라서 스트림 데이터의 점진적 처리를 위한 기법이 필요하며 분산 환경에서 점진적 처리를 위해 노드의 실시간 부하와 성능을 고려한 잡 스케쥴링 기법이 요구된다.

Ⅲ. 제안하는 기법

1. 전체 처리 과정

기존 연구에서 스트림 데이터의 타임 윈도우가 겹치는 경우 중복 처리를 하며 노드의 부하와 성능을 고려하지 않은 스케쥴링 기법으로 실시간 데이터의 빠른 처리를 기대하기 어렵다는 문제점을 가지고 있다. 따라서 스트림 데이터의 실시간 처리를 보장하기 위해 노드의

부하와 성능을 고려한 점진적인 처리 기법이 요구된다. 점진적인 처리를 통해 중복 처리를 피하고 노드의 실시간 부하와 성능을 고려하여 분산 환경에서 데이터의 효과적인 분배와 처리로 실시간 처리를 보장한다.

본 논문에서는 분산 환경에서 스트림 데이터의 실시간 처리 및 분석을 위한 인메모리 기반 점진적인 처리 기법을 제안한다. 제안하는 기법은 스트림 데이터가 입력되면 해시 키 값을 생성하고 마스터 노드에 데이터를 저장한 후 분산 노드에 데이터의 처리를 위해 할당된다. 노드에서 데이터가 처리되면 처리된 결과를 메모리에 저장한다. 이후 데이터가 입력되면 해시 키 값 비교를 통해 동일한 데이터의 경우 메모리에서 처리 결과를 가져와 재사용하며 처리 결과가 저장되어있지 않은 경우 데이터를 노드에 할당하여 처리하고 데이터를 메모리에 저장한다. 데이터의 처리 결과를 재사용하여 중복 처리를 피해 속도를 감소시킨다. 노드의 실시간 부하와 성능을 고려한 스케쥴링 기법을 사용한다. 재사용 가능한 노드를 확인하고 재사용가능한 노드에 우선 데이터를 할당한다. 다음으로 작업 지연을 계산하여 작업이 많은 노드를 피하고 데이터 처리 시간 계산을 통한 노드의 성능을 고려하여 데이터를 할당한다. 데이터의 점진적인 질의 처리와 실시간 부하 및 성능을 고려한 스케쥴링 기법으로 기존 연구보다 데이터의 처리 속도를 감소시킨다.

[그림 1]은 제안하는 기법의 전체 처리 과정을 나타낸 것이다. 스트림 데이터가 입력되면 각 노드의 부하와 성능을 계산한다. 작업 부하가 많아 지연 될 가능성이 있는 노드를 피하고 성능이 좋은 노드에 데이터에 우선적으로 데이터를 할당하여 데이터를 처리한다. 데이터를 할당할 때는 노드의 부하와 성능, 재사용할 데이터를 고려하는 잡 스케쥴링 기법을 이용하여 데이터를 할당한다. 각 노드에 데이터가 할당되면 기존에 처리된 결과가 메모리에 저장되어 있는지 확인한다. 메모리에 기존에 처리된 결과가 저장되어 있다면 처리 결과를 메모리에서 재사용하여 데이터의 중복 처리를 피한다. 기존에 처리된 결과가 메모리에 존재하지 않는다면 데이터는 노드에서 처리하며 각 노드에서 처리된 결과는 병합되어 최종 출력된다.

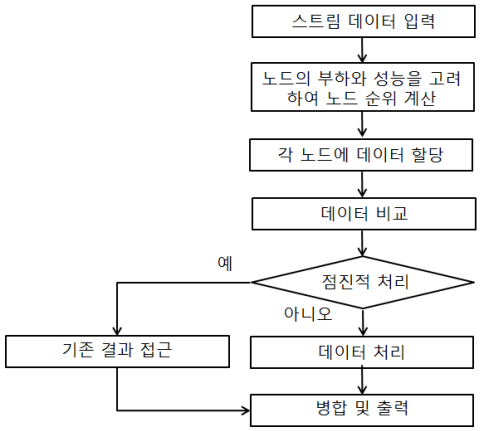
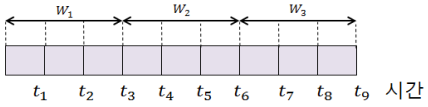


그림 1. 전체 처리 과정

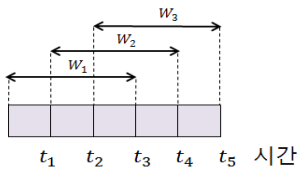
2. 스트림 데이터 처리

스트림 데이터는 RFID, 센서, SNS 등 다양한 곳에서 실시간으로 발생되며 이러한 데이터는 시간 단위의 분석을 필요로 한다. 본 논문에서 <키, 값>으로 구성된 스트림 데이터를 분석하며 키에는 시간, 값에는 해당 데이터 값이 입력된다. 입력된 스트림 데이터는 윈도우로 단위로 처리하기 위해 시간 단위로 구분한다. 일정 시간 간격으로 데이터가 입력되거나 특정 시간에 데이터가 입력되면 그 시간 단위로 윈도우를 설정 한다. 입력된 데이터는 대기 큐에 임시 저장되고 이후 데이터는 잡 스케줄러에 의해 분산 노드에 할당되어 처리된다.

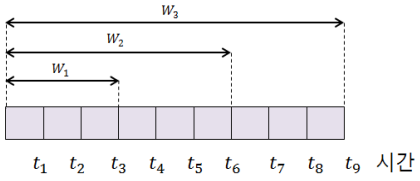
본 논문에서는 입력 데이터를 세 가지 유형으로 정의한다. 질의 타입1은 일반적인 스트림 질의 타입으로 입력되는 데이터를 특정 시간 단위로 분석한다. 질의 타입2는 일부 윈도우가 중복되는 질의 타입으로 윈도우의 중복이 발생할 경우 사용된다. 질의 타입3은 전체 윈도우가 중복되는 질의 타입이다. 하루, 일 년 등의 전체 통계 분석과 같이 처음부터 실시간으로 들어오는 데이터를 모두 처리하여 분석할 때 사용될 수 있다.



(a) 질의 타입 1



(b) 질의 타입 2



(c) 질의 타입 3

그림 2. 질의 타입

스트림 데이터가 $[t_i, t_{i+j}]$ 시간 간격의 윈도우로 입력될 때, i는 초기 데이터 입력시간, j는 데이터 간격을 나타낸다. 예를 들어 [그림 2]의 (a) 질의 타입1은 윈도우가 겹치지 않는 일반적인 스트림 처리를 수행하여 분석하는 것이다. 예를 들어, 1분 간격으로 입력되는 데이터에서 3분단위로 데이터를 분석한다. 계속 입력되는 과정에서 $[t_1, t_3]$, $[t_4, t_6]$, $[t_7, t_9]$ … 의 데이터를 계속 분석하는 과정은 일반적인 스트림 처리가 가능하다. 하지만 겹침이 발생하는 경우는 점진적인 처리 기법이 필요하며 이에 따른 잡 할당 기법이 요구된다.

[그림 2]의 (b) 질의 타입2는 윈도우의 겹침이 발생하는 경우이며 실시간 스트림 데이터를 처리하고 분석하는 과정에서 점진적인 작업이 필요하다. 슬라이딩 윈도우에서 겹침 현상이 발생할 경우, 재계산을 피하기 위해 기존에 처리된 결과를 저장하고 재사용한다. 계속 데이터가 입력되는 과정에서 $[t_k, t_{i+j}]$ 시간의 겹침이 발생할 경우 (k는 이전 데이터와 겹치게 되는 초기 데이터 시간을 나타냄) 겹침이 발생한 부분은 메모리에 저장하기 때문에 재계산이 필요 없이 메모리에서 가져올 수 있다. $[t_1, t_3]$, $[t_2, t_4]$, $[t_3, t_5]$ … 의 시간간격으로 스트림 데이터를 분석하는 경우 2분의 시간 겹침이 발생한다. 처음 $[t_1, t_3]$ 시간의 데이터를 처리하고 처리 결과를 메모리에 저장한다. 다음 $[t_2, t_4]$ 데이터가 입력되면 t_2와 t_3은 처리 결과를 가지고 있기 때문에 별도로 처리할 필요 없이 t_4의 데이터만 처리하면 된다. 마찬가지로 $[t_3, t_5]$의 데이터 입력이 발생하면 t_5의 데이터만 처리하면 된다. 이렇게 2초의 데이터는 재계산 할 필요 없이 처리된 데이터를 불러오고 나머지 1초의 데이터만 처리하기 때문에 전체 데이터 처리 속도는 감소시킨다.

[그림 2]의 (c) 질의 타입3은 이전 윈도우의 전체가 겹치는 경우에 해당하며 겹친 윈도우는 재사용하며 새로 입력되는 데이터만 처리가 필요한 경우이다. 3분단위로 추가적인 연산이 필요한 경우 $[t_1, t_3]$, $[t_1, t_6]$, $[t_1, t_9]$ … 의 데이터 분석의 경우는 두 번째와 마찬가지로 데이터를 메모리에 저장하고 필요한 부분만 계산하여 데이터 처리 속도를 감소시킬 수 있다.

세 가지의 질의 타입 가운데 점진적인 처리가 요구되는 질의 타입은 질의 타입2와 질의 타입3이다. 질의 타입2와 질의 타입3의 경우에는 윈도우의 일부 또는 전부가 겹쳐 중복 처리가 발생하기 때문에 기존 데이터 처리 결과를 저장하고 재사용하는 점진적인 질의 처리가 필요하다. 일반적인 데이터 처리에서는 중복이 발생할 경우 과거에 처리된 결과를 저장하지 않기 때문에 중복 처리한다. 중복 처리를 피하여 데이터의 처리 속도를 향상시키기 위해서 데이터 처리 결과의 재사용이 필요하다. 데이터 처리 결과의 재사용을 통해 데이터 처리 속도를 향상 시킬 수 있으며 이로 인해 실시간 처리가 가능하다.

제안하는 기법의 처리과정은 다음과 같다. 처음으로 스트림 데이터가 입력되면 해시 키 값을 생성하며 임시 큐에 <키, 값> 형태로 저장된다. 이때 입력 데이터의 키는 해시 키 값을 생성하며 데이터는 값으로 저장한다. 그 후 마스터 노드에 있는 데이터와 해시 키를 통해 비교 과정을 거쳐 같은 데이터가 존재하면 메모리에서 처리 결과를 불러온다. [그림 3]은 입력된 데이터의 비교 과정이다. 데이터가 입력되면 해시 키 값을 생성한 후 큐에 <키, 값> 형태로 저장한다. [그림 3]과 같이 처음 입력된 데이터의 해시 키 값은 3d로 시작되며 다음 입력 데이터의 해시 키 값은 45로 시작된다. 큐에 데이터를 저장한 이후 마스터 노드에서 해시 키 값 비교를 통해 과거 처리 결과가 저장되어 있는 노드를 확인

한다. 3d로 시작하는 해시 키 값의 처리 결과는 1번 노드에 저장되어 있으며 45로 시작하는 해시 키 값의 처리 결과는 4번 노드에 저장되어 있음을 확인할 수 있다. 노드의 정보를 파악한 후 해당 노드에 접근하여 데이터를 불러와 처리하는데 이용한다. 이를 통해 데이터 점진적인 데이터 처리가 가능하다.

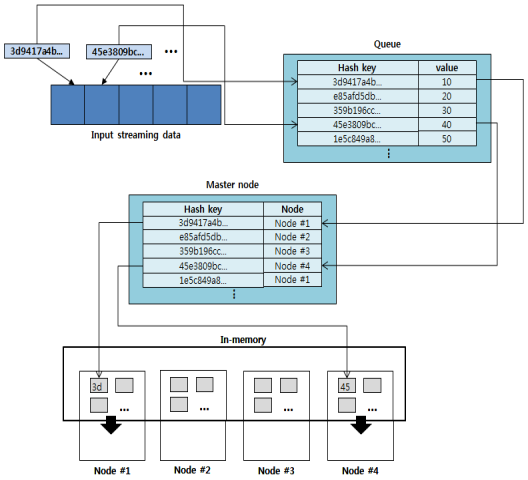


그림 3. 데이터 비교 과정

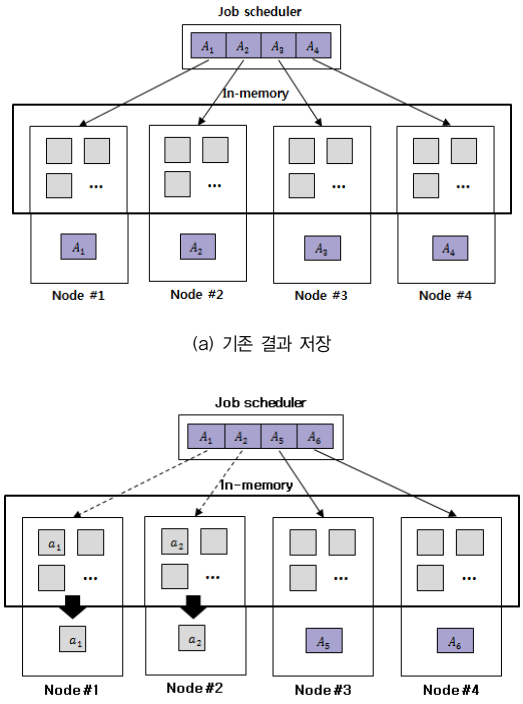


그림 4. 데이터 재사용

점진적 데이터 처리 방법은 기존에 처리된 결과를 메모리에 저장하고 재사용한다. 이를 위해 데이터가 처리되면 처리 결과를 메모리에 저장하고 이후 데이터가 입력되었을 때 메모리에서 처리 결과를 불러와 재사용한다. [그림 4]는 데이터의 점진적 처리 과정을 나타낸다. [그림 4]의 (a)와 같이 처음에 데이터 A_1, A_2, A_3, A_4가 입력되었을 때 각 노드에 데이터가 할당되어 데이터를 처리한다. A_1, A_2의 처리 결과를 각각 a_1, a_2라고 한다면, 데이터의 처리 결과를 재사용하기 위해 메모리에 이 값을 저장한다. 그 후 데이터 A_1, A_2, A_5, A_6이 입력되었을 때, [그림 4]의 (b)와 같이 기존에 처리되지 않았던 A_5, A_6은 노드에 할당되어 처리하며 A_1, A_2 은 처리 결과가 이미 메모리에 저장되어 있기 때문에 재계산하지 않고 처리 결과인 a_1, a_2을 메모리에서 불러온다.

4. 잡 스케쥴링

분산 환경에서 실시간으로 데이터를 처리하기 위해서 점진적인 처리 이외에 각 노드의 부하 및 성능을 고려해야 한다. 기존 점진적인 작업에서 스케쥴링 기법은 노드의 부하와 성능을 고려하지 않아 속도 향상에 어려움이 있다. 따라서 분산 환경에서 노드의 부하와 성능을 고려하여 적절하게 데이터를 분배할 잡 스케쥴링 기법이 필요하다. 제안하는 잡 스케쥴링 기법은 각 노드의 작업 지연을 실시간으로 파악하여 어느 노드에 부하가 발생하는지 확인한다. 또한 노드의 처리 시간을 계산하여 노드의 성능을 파악한다.

제안하는 기법에서는 분산 환경에서 데이터를 처리한다. 분산 노드는 하둡과 같이 한 개의 마스터 노드와 여러 개의 슬레이브 노드로 구성되며 마스터 노드에서는 전체 슬레이브 노드를 관리한다. 마스터 노드는 각 슬레이브 노드의 작업 지연정보를 통해 부하를 계산하

고 처리 시간을 계산하여 노드의 성능을 파악한다. [그림 5]는 마스터 노드와 슬레이브 노드에 대한 예를 나타낸다. 마스터 노드에서는 각 슬레이브 노드의 정보를 가지고 있다. 노드 1에는 처리해야할 작업이 4개가 있으며 노드 2는 1개, 노드 3은 3개, 노드4는 2개의 처리해야할 작업을 가지고 있다. 성능면에서 1번 노드의 성능이 가장 좋은 것을 확인할 수 있으며 4번 노드의 성능이 가장 좋지 않은 것을 확인 할 수 있다. 이러한 정보를 마스터 노드에서 실시간으로 확인한다.

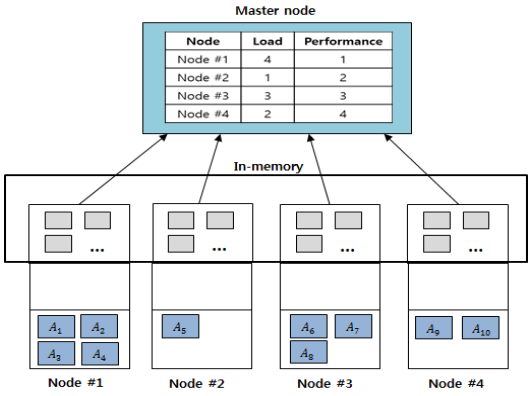


그림 5. 마스터 노드, 슬레이브 노드

처음 데이터가 입력되면 프로세스 사이에 우선순위를 두지 않고 순서대로 시간단위로 CPU를 할당하는 방식인 라운드 로빈 스케쥴링 기법으로 데이터를 분배한 후 실시간으로 노드의 성능과 부하를 확인하여 잡 스케쥴링을 수행한다. 노드의 성능의 경우 데이터가 처리되면 각 노드에서 데이터가 처리되는데 걸리는 평균 시간을 계산하여 노드의 성능을 파악한다. 식 (1)은 이미 완료된 잡의 처리 시간을 이용하여 평균 잡의 처리 시간을 측정하는 식이다. 평균 잡의 처리 시간을 측정하여 이후 잡을 노드에 할당할 때 처리 시간이 빠른 노드에 우선적으로 할당하여 성능을 향상시킨다. i는 노드, k는 잡을 의미하며 AT_i 는 평균 잡 완료 시간, $Ttask_i$는 잡이 처리 되는 시간을 나타낸다.

$$AT_i = \frac{1}{k}\sum_{k=1}^{\max} Ttask[k] \qquad (1)$$

실시간 노드의 부하를 고려한다. 각 노드에 존재하는 잡을 실시간으로 확인하여 어느 노드에 부하가 많은지 파악한다. 노드의 부하가 많은 노드에 잡을 할당한다면 처리 시간이 증가할 것이므로 되도록이면 노드의 작업 부하가 적은 노드에 더 많은 잡을 할당해야 한다. 식 (2)는 i노드에서 할당된 잡의 수를 측정하는 식이다.

$$S_i = \sum_{k=1}^{\max} task[k] \qquad (2)$$

식(3)은 식(1)에서 나타낸 노드의 성능, 식(2)에서 나타낸 실시간 부하를 종합적으로 고려한 식이다. 식(3)의 값을 산출하여 각 노드에 잡을 할당한다. 하지만 노드의 상태나 부하에 따라 α, β 값을 임의로 설정할 수 있으며, 노드의 성능차이가 큰 경우는 α 값을 크게 설정하고, 실시간으로 부하가 생기는 노드가 많다면 β의 값을 크게 설정한다.

$$JS = \alpha AT_i + \beta S_i \quad (\alpha+\beta=1) \qquad (3)$$

[그림 6]은 실시간 노드의 부하를 설명한다. 노드1과 노드 3에 4개의 잡이 처리되지 않은 상태로 있고 노드2에는 1개의 잡, 노드4에는 2개의 잡이 처리되지 않은 상태로 있다. 잡을 할당할 때 노드2에 우선적으로 할당하고 다음으로 노드4에 할당하게 된다.

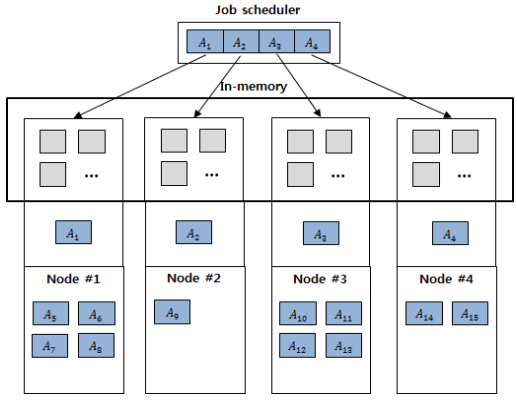


그림 6. 실시간 노드 부하

제안하는 기법에서는 메모리 기반 처리로 휘발성의 문제가 있어 데이터의 복제본을 만들어 놓는다. 이 복제본은 데이터 처리로도 사용될 수 있다. 기존 노드에

부하가 존재 할 경우 기존 노드와 복제본 노드 비교를 통해 어느 곳에서 작업을 수행하는 것이 효율적인지 판단한다. 기존 노드에서의 데이터 처리 시간을 T_a, 복제본 노드에서 데이터 처리 시간을 T_{rep}라고 할 때, 식(4)를 만족할 경우 복제본 노드에서 처리하며 만족하지 않을 경우 기존 노드에서 데이터를 처리한다.

$$T_a < T_{rep} \quad (4)$$

$$T_a = S_{ai} + T \quad (5)$$

$$T_{rep} = S_{repi} + T \quad (6)$$

[그림 7]과 같이 잡 A_1이 노드 1에 할당되고 노드1에는 원본의 결과가 저장되어 있고 노드2에는 복제본이 저장되어 있으면 노드1에서는 부하가 많이 발생하고 있어 노드2에서 처리한다. 노드의 작업 지연을 계산하여 노드의 부하를 파악하고 처리 시간을 계산하여 성능을 고려한 잡 스케쥴링 기법을 통해 분산 노드에 데이터를 효과적으로 분배하여 데이터 처리 시간을 감소시킨다.

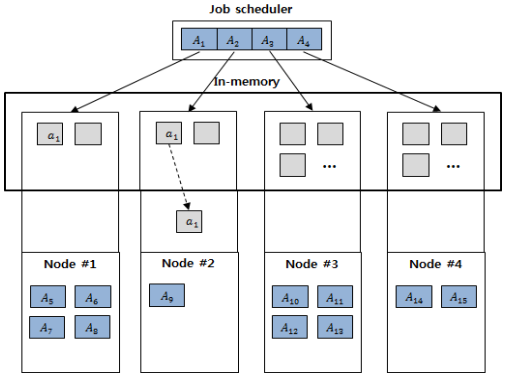


그림 7. 복제본 노드에서 처리

5. 메모리 관리

메모리에 데이터가 가득 차면 더 이상 데이터를 저장할 수 없기 때문에 효과적인 메모리 관리가 필요하다. 일반적으로 실시간 분석에 사용되는 데이터는 최신의 데이터를 중심으로 이루어지기 때문에 과거의 데이터보다는 최근 데이터가 많이 사용될 것이므로 LRU기법을 사용하여 메모리를 관리를 한다. 다양한 논문[12]에서 LRU기법을 채택하고 있으며, LRU기법은 가장 오랫동안 사용되지 않은 데이터를 교체하는 기법으로 과거에 사용했던 데이터 중에서 가장 오래된 데이터 순으로 삭제한다. 메모리에서 지운 데이터는 영구적으로 삭제하는 것이 아니라 디스크에 저장해두어 필요한 경우 가져다 쓸 수 있게 한다.

IV. 성능평가

본 논문에서는 제안하는 기법의 우수성을 보이기 위해 스파크[3]와의 성능평가를 수행하였다. [표 1]은 성능평가 환경을 보여준다. 임의의 스트림 데이터를 생성하고 논문에서 제시한 질의 타입 3가지를 평가하였다. 성능평가 환경은 Intel(R) CoreTM i3 CPU100 프로세서, 4G메모리, 500GB디스크 환경으로 구성된다. Java를 이용하여 4개의 가상 노드에서 성능평가를 수행하였다. 각 질의 타입의 성능평가는 잡의 수가 증가함에 따라 데이터의 처리 시간을 평가하였으며, 한 개의 잡은 100MB의 크기를 가진다.

표 1. 성능평가 환경

구분	내용
프로세서	Intel(R) CoreTM i3 CPU100
메모리	4 GB
디스크	500 GB
프로그램 언어	Java
노드 수	4개

스파크는 가장 대표적인 인메모리 기반 분산 데이터 처리 시스템으로 제안하는 기법과 가장 유사한 질의 처리를 보이고 있기 때문에 성능평가 대상으로 선택하였다. 질의 타입은 본 논문에서 제시한 질의 타입1, 질의 타입2, 질의 타입3 세 가지를 비교 하였으며 wordcount를 통해 처리 시간을 평가하였다. 또한, 스파크의 스케쥴러와 제안하는 기법의 스케쥴러의 데이터 처리 속도를 비교하는 성능평가를 수행하였다. 제안하는 기법은 점진적인 처리와 잡 스케쥴링 기법을 사용하여 우수한 성능을 보여주는 것을 확인할 수 있다.

[그림 8]은 질의 타입1에 대해 노드에 부여된 잡의 수가 증가함에 따라 데이터의 처리 속도를 스파크와 비교 실험한 결과이다. 질의 타입1은 타임 윈도우가 겹치는 점진적인 처리가 아닌 일반적인 데이터의 처리이며 기존 결과를 재사용하지 않는다. 데이터가 증가함에 따라 속도가 증가하며 기존 기법과 비슷한 결과가 도출되는 것을 확인할 수 있다. 점진적인 처리를 하지 않았음에도 제안하는 기법이 기존 기법보다 약간 빠른 처리를 보이는 이유는 제안하는 스케쥴링 기법을 사용하여 노드의 성능과 부하를 고려했기 때문이다.

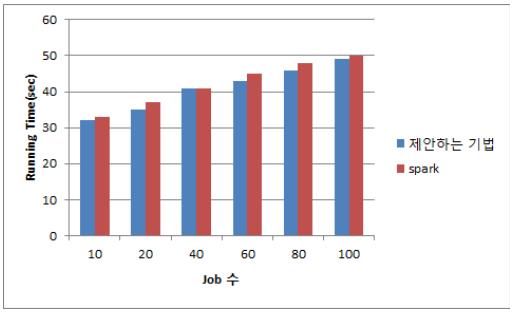


그림 8. 질의 타입1 수행 시간

[그림 9]는 질의 타입2의 데이터 처리 속도를 스파크와 비교 실험하였다. 질의 타입2의 경우는 타임 윈도우가 일부 겹치는 점진적인 데이터 처리이기 때문에 기존 데이터의 처리 결과를 메모리에 저장하고 재사용한다. 데이터 처리 결과의 일부를 재사용하여 데이터 처리를 수행하여 처리 시간이 감소됨을 확인할 수 있다. 제안하는 기법이 스파크에 비해 약 22% 질의 수행 시간이 감소되었다.

[그림 10]는 질의 타입3과 스파크와 질의 수행시간을 비교 실험 결과를 나타낸다. 잡의 수가 증가함에 따라 데이터의 처리 속도를 비교하였으며 질의 타입3의 경우에는 타임윈도우의 전체가 겹치는 질의 타입이다. 과거 데이터 처리 결과를 메모리에 저장하고 재사용하는 과정에서 처리된 결과 데이터 전체를 재사용하기 때문에 가장 많은 중복 처리를 피해 세 가지의 질의 타입 중 가장 우수한 성능을 보여준다. 성능평가 결과 제안하는 기법은 기존 기법에 비해 질의 수행 시간에서 약 39% 우수한 성능을 보이는 것으로 나타났다.

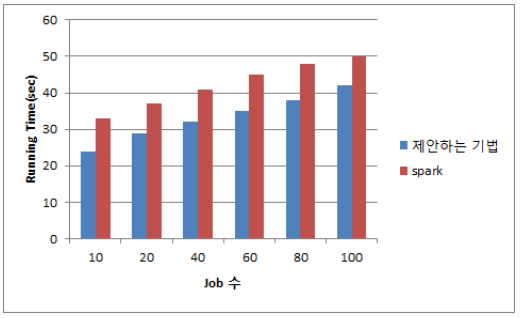


그림 9. 질의 타입2 수행 시간

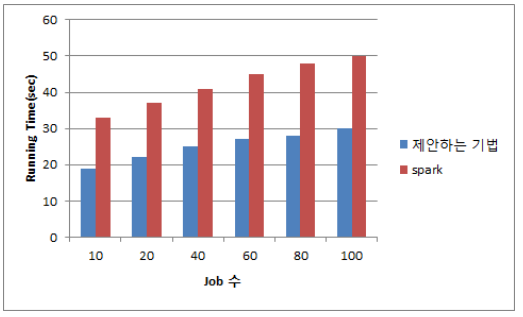


그림 10. 질의 타입3 수행 시간

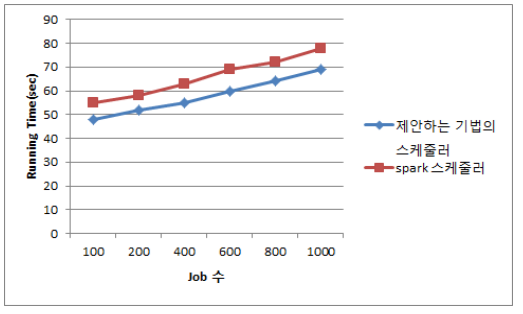


그림 11. 스케쥴러 수행 시간

[그림 11]은 잡의 수가 증가함에 따라 제안하는 기법의 스케쥴러와 스파크의 스케쥴러의 수행시간을 비교 실험한 결과이다. 제안하는 기법은 각 노드의 작업 지연을 계산하여 부하가 많은 노드를 피해 데이터를 할당하고 데이터 처리 시간을 계산하여 노드의 성능 파악하하고 데이터를 할당한다. 제안하는 스케쥴링 기법은 스파크의 스케쥴러에 비해 수행 시간이 감소됨을 확인하였다. 성능평가 결과 제안하는 기법의 스케쥴러가 스파

크의 스케줄러에 비해 약 11% 질의 수행 시간이 감소되었다.

V. 결론

본 논문에서는 인메모리 기반 스트림 데이터의 점진적 처리 기법을 제안하였다. 제안하는 기법은 과거에 처리된 데이터를 메모리에 저장하고 실시간으로 데이터가 입력되었을 때 해시 키 값으로 비교하여 기존에 처리 된 데이터 중 필요한 데이터를 활용하여 처리한다. 또한 노드의 성능과 저장된 데이터를 고려한 잡 스케줄링 기법을 수행한다. 성능평가 결과 제안하는 기법이 스트림 데이터 처리에서 약 39% 우수한 성능을 보이는 것으로 나타났다. 추후 연구로는 메모리에 저장된 처리 결과의 업데이트 기법을 연구할 것이다.

참 고 문 헌

[1] J. Dean and S. Ghemawat, "MapReduce: simplified data processing on large clusters," Proc. conference on Symposium on Operating Systems Design & Implementation, pp.137-150, 2004.

[2] Matei Zaharia, Mosharaf Chowdhury, Tathagata Das, Ankur Dave, Justin Ma, Murphy McCauly, Michael J. Franklin, Scott Shenker, and Ion Stoica, "Resilient Distributed Datasets: A Fault-Tolerant Abstraction for In-Memory Cluster Computing," NSDI pp.15-28, 2012.

[3] https://storm.apache.org/

[4] D. Tiwari and Y. Soligin, "MapReusing Computation in an In-Memory MapReduce System," Proc. International Parallel and Distributed Processing Symposium, pp.61-71, 2014.

[5] Pramod Bhatotia, Marcel Dischinge, Rodrigo Rodrigues, and Umut A Acar, "Slider: Incremental Sliding-Window Computations for Large-Scale Data Analysis," Middleware, pp.61-72, 2014.

[6] Fan Zhang, Junwei Cao, Samee U. Khan, Keqin Li, and Kai Hwang, "A task-level adaptive MapReduce framework for real-time streaming data in healthcare application," Future Generation Computer System, pp.149-160, 2015.

[7] Doug Laney, *3D data management: Controlling data volume, velocity, and variety,* Technical report, META Group, 2001.

[8] 이미영, 최완, "빅데이터 분석을 위한 빅데이터 처리 기술 동향," 정보처리학회지, 제19권, 제2호, pp.20-28, 2012.

[9] 김현규, 강우람, 김명호, "중첩 윈도우를 가진 데이터 스트림을 위한 효율적인 조인 알고리즘," 정보과학회논문지, 제15권, 제5호, pp.365-369, 2012.

[10] 이욱현, "스트림 데이터에서 회귀분석에 기반한 빈발항목 예측," 한국콘텐츠학회논문지, 제9권, 제1호, pp147-158, 2009.

[11] 김재인, 김대인, 송명진, 한대영, 황부현, "다차원 스트림 데이터 환경에서 이벤트 가중치를 고려한 시간 관계 탐사," 한국콘텐츠학회논문지, 제10권, 제2호, pp.99-110, 2011.

[12] S. Chandrasekar, R. Dakshinamurthy, P. G. Seshakumar, B. Prabavathy, and Chitra Babu, "A Novel Indexing Scheme for Efficient Handling of Small Files in Hadoop Distributed File System," International Conference on Computer Communication and Informatics, pp.1-8, 2013.

저 자 소 개

복 경 수(Kyoungsoo Bok) 종신회원

- 1998년 2월 : 충북대학교 수학과 (이학사)
- 2000년 2월 : 충북대학교 정보통신공학과 (공학석사)
- 2005년 2월 : 충북대학교 정보통신공학과 (공학박사)

- 2005년 3월 ~ 2008년 2월 : 한국과학기술원 정보전자연구소 Postdoc
- 2008년 3월 ~ 2011년 2월 : 가인정보기술 연구소 연구원
- 2011년 3월 ~ 현재 : 충북대학교 전자정보대학 정보통신공학부 초빙교수

<관심분야> : 데이터베이스 시스템, 이동객체 데이터베이스, 소셜 네트워크, 빅데이터 등

육 미 선(Misun Yook) 준회원

- 2014년 2월 : 충북대학교 정보통신공학과(공학사)
- 2014년 3월 ~ 현재 : 충북대학교 정보통신공학과 석사과정

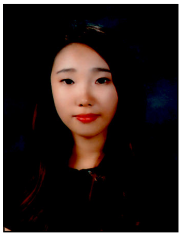
<관심분야> : 데이터베이스 시스템, 빅 데이터, 소셜 네트워크 서비스 등

노 연 우(Yeonwoo Noh) 준회원

- 2014년 2월 : 충북대학교 정보통신공학과(공학사)
- 2014년 3월 ~ 현재 : 충북대학교 정보통신공학과 석사과정

<관심분야> : 데이터베이스 시스템, 빅 데이터 등

한 지 은(Jieun Han) 준회원



- 2014년 2월 : 충북대학교 정보통신공학과(공학사)
- 2014년 3월 ~ 현재 : 충북대학교 정보통신공학과 석사과정

<관심분야> : 빅 데이터, 프러버넌스 데이터, RDF 등

김 연 우(Yeonwoo Kim) 정회원

- 2002년 2월 : 청주대학교 법학과(법학사)
- 2012년 2월 : 충북대학교 정보통신공학(공학석사)
- 2012년 3월 ~ 현재 : 충북대학교 정보통신공학과 박사과정

<관심분야> : 데이터베이스 시스템, 소셜 네트워크 서비스, 빅데이터

임 종 태(Jongtae Lim) 정회원

- 2009년 2월 : 충북대학교 정보통신공학과(공학사)
- 2011년 2월 : 충북대학교 정보통신공학과(공학석사)
- 2015년 8월 : 충북대학교 정보통신공학과(공학박사)

<관심분야> : 데이터베이스 시스템, 시공간 데이터베이스, 위치기반 서비스, 이동 P2P 네트워크, 소셜 네트워크 서비스, 빅 데이터 등

유 재 수(Jaesoo Yoo) 종신회원

- 1989년 2월 : 전북대학교 컴퓨터공학과(공학사)
- 1991년 2월 : 한국과학기술원 전산학과(공학석사)
- 1995년 2월 : 한국과학기술원 전산학과(공학박사)
- 1995년 3월 ~ 1996년 8월 : 목포대학교 전산통계학과 전임강사
- 1996년 8월 ~ 현재 : 충북대학교 전자정보대학 교수

<관심분야> : 데이터베이스 시스템, XML, 멀티미디어 데이터베이스, 분산 객체 컴퓨팅, 빅 데이터 등