한국지능시스템학회 논문지 2009, Vol. 19, No. 4, pp. 485-492

분산 모바일 서비스의 다중 스트리밍을 위한 가변 클러스터링 관리

Variable Clustering Management for Multiple Streaming of Distributed Mobile Service

정택원 · 이종득*

TaegWon Jeong and ChongDeuk Lee*

전북대학교 응용시스템공학부 전자통신공학전공

요 약

모바일 서비스 환경에서 시간 동기화에 의해 생성된 패턴들은 데이터 스트리밍으로 인하여 인스턴스 값들이 다르게 스트리밍 된다. 본 논문에서는 유연한 클러스터링을 지원하기 위해 가변클러스터링 관리 기법을 제안하며, 이 구조는 다중 데이터 스트리밍을 동적으로 관리하도록 지원한다. 제안되는 기법은 일반적인 스트리밍기법과 달리 데이터 스트림 환경에서 동기화를 효율적으로 지원하는 기능을 수행하며, 구조적 표현단계와 적합성 표현단계를 거쳐 클러스터링 스트리밍이 관리된다. 구조적 표현 단계는 레벨정합과 누적정합을 수행하여 스트림 구조가 표현되며, 동적 세그먼트와 정적세그먼트 관리를 통해서 클러스터링 관리가 가변적으로 수행되도록 하였다. 제안된 기법의 성능 평가를 위해서 k-means 기법, C/S 서버 기법 그리고 CDN 기법과 시뮬레이션평가를 수행하였으며 그 결과 제안된 기법의 성능이 효율적임을 알 수 있었다.

키워드 : 데이터 스트리밍, 가변클러스터링, 레벨정합, 세그먼트

Abstract

In the mobile service environment, patterns generated by temporal synchronization are streamed with different instance values. This paper proposed a variable clustering management method, which manages multiple data streaming dynamically, to support flexible clustering. The method manages synchronization effectively and differently with conventional streaming methods in data streaming environment and manages clustering streaming after the structural presentation level and the fitness presentation level. In the structural presentation level, the stream structure is presented using level matching and accumulation matching, and clustering management is carried out by the management of dynamic segment and static segment. The performance of the proposed method is tested by using k-means method, C/S server method, CDN method, and simulation. The test results showed that the proposed method has better performance than the other methods.

Key Words : Data streaming, variable clustering, level matching, segment

1. 서 론

최근에 모바일 서비스 기술은 디지털 기술과 정보통신 기술의 발달로 인하여 유/무선, 모바일 기기 등이 융합되어 사용자에게 실시간으로 서비스를 제공하는 형태로 빠르게 발전하고 있다. 이 같은 추세는 다양한 모바일 서비스에 대한 수요를 증가시키고 있으며 신뢰성 있는 서비스를 제공하기 위해 가변 데이터 클러스터링 관리 기법이 요구되고 있다[1, 2, 3]. 모바일 환경에서 데이터 스트림을 위한 방법은 요약기법[2], 샘플링과 통계적 기법[4], 클러스터링[5, 6], 데이터 분류 기법[7, 8] 등 여러 가지 기법들이 제안되고 있다. 모바일 환경에서 데이터 스트림을 위한 데이터 크기와 타임스탬프(Time stamp)[1]는 스트림에 중요한 영향을 미치게 되며, 클러스터링은 저장공간 탐색, 동기화 및 실행시간 등에 영향을 미치게 된다. 또한 모바일 서비스의 제한된 대역폭과 빈번한 끊김 현상으로 인하여 모바일 트랜잭션 처리 시에 QoS와 신뢰성이 떨어지는 문제가 발생되고 있다[9]. 이러한 문제를 해결하기 위해 클러스터링 관리기법[3] 등이 제안되고 있으며, 본 논문에서는 다중 데이터를 스트리밍하기 위해 가변 클러스터링 관리기법을 제안한다. 스트리밍이 수행될 때 타임스탬프와 데이터 포인터는 동시에 도착되며, 데이터 포인터는 이전 포인터와 매우 높은 상관관계를 가진다. 이러한 방법을 수행하기 위해서 가변길이와 고정길이의 세그먼트들을 분석하여 1-단계로 데이터 스트리밍을 위한 계층적 구조로의 환경 분석과정과 2-단계로 구조화된 클러스터링에서 동적인 상황인식 데이터 스트리밍이 수행되도록 질의 확장을 수행한다. 데이터 스트림 환경에서 서로 다른 시간에 생성된 스트리밍 패턴들은 데이터 변화가 동적으로 수행되며 규칙과 패턴에 따른 변화의 빈도는 클러스터링에 따라 다르게 발생한다. 예를 들어 인접 노

접수일자 : 2009년 1월 20일
완료일자 : 2009년 7월 9일
* 교신 저자

드들로부터 수집된 스트림 데이터들은 항상 같은 클러스터들을 유지할 수 있으며, 시간이 흘러도 거의 변하지 않는다. 그러나 멀티미디어 정보와 같은 경우에는 데이터 크기로 인하여 같은 클러스터를 유지할 수 있으나 상황에 따라 다른 클러스터를 유지할 수 있다. 따라서 이러한 데이터 스트리밍의 경우에는 클러스터들을 상황 변화에 따라 가변적으로 관리하고 스트리밍 할 수 있도록 구성해야 한다. 그리고 스트리밍과 클러스터링은 서비스 상황에 따라 변할 수 있다. 이처럼 구조화된 클러스터링 관리는 데이터 볼륨이 커지고 플로우가 동적일 때 효과적이다. 따라서 본 논문에서는 시간적 상황 인식에 따른 가변클러스터링 관리 기법을 제안함으로서 저장 공간과 탐색공간의 낭비를 줄이는 효과를 가져오도록 하며, 검색을 효율적으로 지원하여 비용 문제의 낭비를 줄이는 효과를 가져오도록 하게 한다. 본 논문의 구성은 다음과 같다. 2장에서는 제안된 기법에 대한 관련 연구에 대해서 살펴보며, 3장에서는 가변적 클러스터링 관리 모델에 대해서 살펴본다. 그리고 4장에서는 제안된 시스템의 성능평가에 대해서 살펴보며, 끝으로 결론에 대해서 살펴본다.

2. 관련연구

스트리밍은 모바일 및 무선 인터넷상에서 주문형 서비스를 제공하기 위한 중요한 서비스 기법으로 자리를 잡아가고 있다. 분산 모바일 환경의 스트리밍 서비스를 수행하기 위하여 많은 기법들이 연구되고 있으며 이중에서 클라이언트-서버구조화 기법, CDN(Content Distribution Networks) 기법 그리고 P2P(Peer To Peer)기반의 기법이 주로 연구되고 있다[10, 11]. 클라이언트 구조화 기법은 배칭(batching), 패칭(patching), 그리고 주기적 방송 기법(periodic broadcasting)에 기반을 둔 기법[12]으로서 이 기법은 IP 네트워크상의 멀티캐스트 기능 부족으로 인하여 구현상에 문제가 발생되고 있으며, 대규모의 스트리밍 서비스에는 적합하지 않는 문제점을 가지고 있다. CDN 기법은 인터넷상의 CDN 서버의 수에 기반을 둔 기법으로서 스트리밍 서비스를 수행할 스트림 프레임이 먼저 서버에 분산되어 있어야 하고 이들 각 서버를 통하여 이웃한 클라이언트에 프레임이 전송될 수 있어야 한다. 이 기법은 광역의 전송 구조에는 많은 비용이 요구되며 멀티미디어 서비스를 수행하기 위해서는 많은 자원이 요구된다는 문제점을 가지고 있다. 따라서 한정된 자원으로 인하여 동시에 많은 데이터 서비스가 요구되는 멀티미디어 서비스에는 한계가 있다. P2P 기반의 서비스 기법[13]에서 클라이언트는 피어(peer)로서 다른 클라이언트에게 서비스를 받을 뿐만 아니라 서비스를 제공하기도 한다. 이때 다른 클라이언트와 서로 협력적 관계를 통하여 서버 부담을 줄여주어야 하며, 이와 같은 서버 부담을 통하여 시스템의 성능을 개선하는 효과를 가져올 수 있다. 그러나 분산 환경에서 분포된 많은 자원으로 인하여 탐색 공간 및 저장공간 그리고 네트워크 대역폭과 같은 문제가 발생되고 있다. 또한 클라이언트-서버 구조화 기법 및 CDN 시스템에 비해서 신뢰성이 떨어지고 있으며, 품질보증이 떨어진다는 단점을 가지고 있다.

3. 제안된 가변 클러스터링 관리 모델

클러스터링으로 구조화 된 데이터 스트림들은 사용자의 주문에 따라 서비스가 다르게 수행되어야 하며, 스트리밍 서비스 또한 세그먼트 크기에 따라 클러스터들이 다르게 구성되어야 한다. 그러나 상황에 따라 동적으로 스트리밍 되는 연속적인 데이터 스트림들을 세그먼트 크기에 따라 클러스터링을 구성하는 일은 매우 어려운 일이다. 이러한 이유는 원시적인 연속적 스트림 데이터를 저장하기 위한 메모리의 부족현상과 스트림들을 상황적으로 관리하는 기능이 미흡하기 때문이다. 이와 같은 문제점을 개선하기 위하여 거리 기반의 임계값[1]을 이용한 방법이 제안되고 있으며 이 방법은 세그먼트 크기에 따른 임계값을 찾는 방법이다. 이 장에서는 클러스터 관리기에서 클러스터링 환경을 분석하여 이를 구조화하는 구조적 표현단계와 이를 통해서 질의를 수행하는 선언적 표현단계로 구성되며, 제안된 시스템 구조는 그림1과 같다.

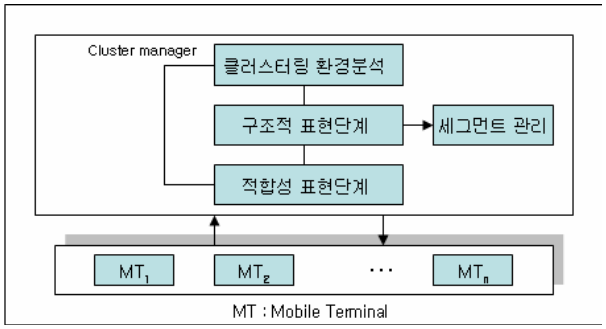

그림 1. 제안된 시스템 구조
Fig. 1. The proposed system structure

3.1 클러스터링 환경 분석

클러스터링 환경 분석 단계는 데이터 스트림을 계층구조로 표현하는 단계를 말하며, 구조적 표현단계와 상호 작용을 통해서 세그먼트를 관리하게 된다. 구조적 표현 단계에서 구성된 클러스터링에 선언적 표현단계와 상호 작용을 통하여 세그먼트를 유연하게 스트리밍하게 된다. 예를 들어 특정 사용자가 선호하는 웹 서비스를 1년 동안의 클러스터링을 알아보기 위하여 매월 세그먼트의 크기를 1개월 단위로 하여 12개의 클러스터를 만들 수 있다. 데이터 스트림 환경에서 제한된 도메인에 의한 원시 데이터 스트림은 한번만 파싱되고 이후에는 파싱되지 않는다. 따라서 이러한 초기 파싱만을 파악하여 파싱을 수행하도록 하며, 본 논문에서 수행되는 구조적 표현 단계에 의한 스트리밍은 그림2에서처럼 초기 스트림 과정에서 스트리밍이 수행되는 것이 아니라 타임스탬프에 의한 통계적 기법이 적용된 이후 스트리밍이 수행된다. 타임스탬프가 수행될 때 각각의 스트림은 새로운 값을 가지게 되며, 이때 전체적으로 n개의 데이터 스트림이 생성된다. 따라서 타임스탬프가 ts이고 n 개의 stream이 $\{s_1, s_2, \ldots, s_n\}$이면, i-번째 스트림 값 $sv_i=\{sv_i^1, sv_i^2, \ldots, sv_{n_i}^k\}$이다. 여기서 sv_i^j 는 임의의 t시간에 도착되는 j번째 블록 세그먼트의 스트림 값이다.

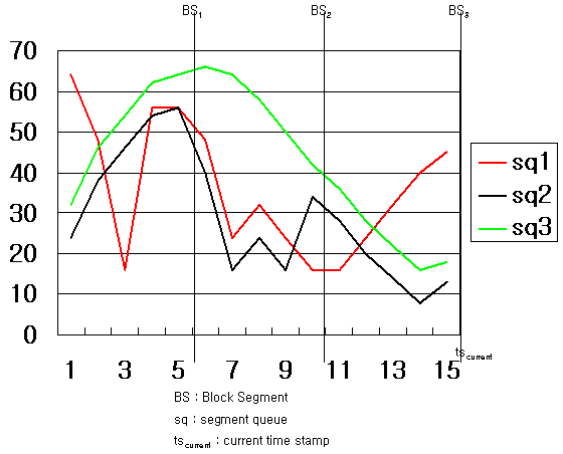

그림 2. 초기 스트림
Fig. 2. Initial stream

3.1.1 스트림 클러스터링

스트림 클러스터링을 수행하기 위해서는 초기 스트림구조로부터 서브 스트림들을 추출하게 된다. 예를 들어 p는 클러스터의 수라 하고, BS(Block segment)는 타임스탬프가 $ts_{current}$일 때 클러스터링을 위한 블록 세그먼트 크기라 하자. 이때 초기 스트림의 p개의 클러스터에서 t개의 블록 세그먼트 BS를 위한 스트리밍 생성 과정은 $1 \le i \le t$일 때 $create(BS_i)=\{create_1(bs_i), create_2(bs_i), \ldots, create_p(bs_i)\}$이 되며, 각 구간에서 서브스트림을 최소화하기 위한 각각의 블록 세그먼트 생성비용은 다음과 같이 정의된다.

(정의1) $Cost(create(bs_i))$는 $[(ts_{current}-bs\times i, ts_{current}-bs\times(i-1)]$이다.

그리고 $create_j(bs_i)$가 다음과 같은 성질, 즉 $\bigcap_{j=1}^{p} create_j(bs_i)=\varnothing$이고, $\bigcup_{j=1}^{p} create_j(bs_i)=\{s_1(bs_i), s_2(bs_i), \ldots, s_n(bs_i)\}$이면, 클러스터링에 대한 스트리밍의 결정은 다음과 같이 정의된다.

(정의2) $s_q(bs_i)=\{stream_q^{(ts_{current}-BS\times i)+1}, stream_q^{(ts_{current}-BS\times i)+2}, \ldots, stream_q^{(ts_{current}-BS\times i)+p}\}$이다. 여기서 q는 $1 \le q \le n$이며, j번째의 클러스터에서의 스트리밍이 수행되는 블록 세그먼트이다. 그리고 $stream_q^{(ts_{current}-BS\times i)+p}$는 임의의 BS(Block Segment)에서 질의를 수행한 후 생성된 스트림들이다. 예를 들어 그림2에서처럼 $ts_{current}$=15이고, 3개의 데이터 스트림 대기 큐가 $\{sq_1, sq_2, sq_3\}$가 그림 3과 같다고 하자.

스트리밍을 위한 클러스터링 질의를 p=2, BS=5, t=2라 하면 스트리밍을 위한 클러스터링 $create(bs_1)=\{create_1(bs_1), create_2(bs_1)\}$과 $create(bs_2)=\{create_1(bs_2), create_2(bs_2)\}$가 생성된다. 따라서 bs_1에 대한 클러스터링 스트리밍은 $create_1(bs_1)=\{s_1\}$과 $create_2(bs_1)=\{s_2, s_3\}$가 된다.

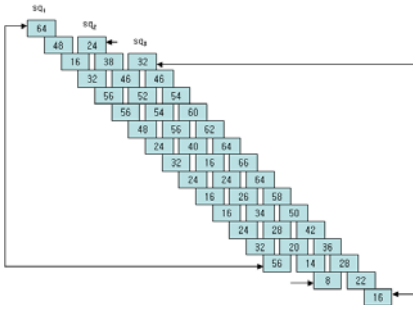

그림 3. sq_1, sq_2, sq_3에 대한 대기 큐
Fig. 3. Queue for sq_1, sq_2 and sq_3

이것은 (정의1)에 의해 즉 $\{(15-5)\times1, 15-5\times(1-1)\}=\{(10, 15)\}$의 구간에서 클러스터링스트리밍이 수행됨을 의미한다. 그리고 bs_2의 경우에는 $\{(15-(5\times2), 15-5\times(2-1)\}=\{(5, 10)\}$의 구간에서 클러스터링 스트리밍이 수행됨을 의미한다.

3.1.2 BS에 의한 클러스터링

일반적으로 계층구조화 된 스트림들은 여러 해를 가질 수 있으며, 이 절에서는 BS의 크기에 따라 클러스터링을 수행하는 방법에 대해서 살펴본다. BS의 크기에 따라 스트리밍을 수행하기 위해서는 연속적인 데이터 스트림들이 가변적으로 클러스터링을 수행할 수 있어야 하며 이것은 원시적인 연속적 스트림 데이터를 저장하기 위한 메모리의 부족현상을 능동적으로 관리하기 위함이며 스트림들을 가변적으로 수행하기 위한 것이다. 이와 같은 기법을 위해서 거리기반의 임계값을 이용하고 있으나 이 방법은 BS 크기에 따른 임계값을 찾기가 어려운 문제점이 있다. 본 논문에서는 스트림 구조를 이용하여 초기 스트림구조에서 다중 데이터 스트리밍이 가능하도록 스트림을 구조화한다. 이렇게 구조화된 스트림 구조는 데이터 스트림의 수가 증가될 때 스트리밍을 효율적으로 제공하며, 계층구조에서의 다중 스트림 레벨에 대한 스트리밍 서비스를 가변적으로 수행하게 된다. 스트림 계층구조 stream(h)는 타임스탬프 구간 $ts=[t_s, t_e]$(여기서 t_s는 구간이 시작되는 시간, t_e는 끝나는 시간) 일 때 초기 서브스트림들을 분석하여 수행하며, 각 레벨을 구성하는 모델은 먼저 (L-1)레벨을 수행한 후 캐싱 블록 CB_h를 생성한다. 그리고 레벨 L의 구성은 가장 최근의 데이터 포인터에 타임스탬프를 찍어 구성하며, 구간에서 타임스탬프 레벨 ts_e^L이 끝날 때까지 반복한다. 그림4는 CB_t=T, CB_h=2, ts=8일 때 다중 스트림 레벨 구조를 나타낸 예이다.

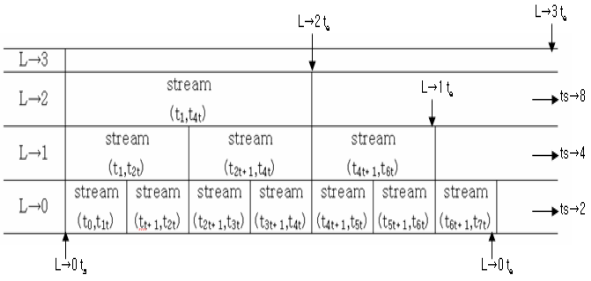

그림 4. 다중 스트림 레벨 구조
Fig. 4. Multi-level stream

다중 스트림을 위한 레벨에서 L→0은 초기 레벨이며, L→1은 1계층, L→2는 2계층, L→3은 3계층을 의미한다. 그리고 레벨의 입력 값은 먼저 임시 캐시 블록에 저장되며, CB_t 값이 도착된 후에 L→0을 생성하고, 생성된 후에는 끝 시간 t_e^0이 갱신되게 된다. 이후 CB_t값은 임시CB에서 본래의 CB로 이동되며, 본래의 스트림 중 가장 최근의 레벨 스트림 α를 유지하게 된다. 새로운 CB_h가 레벨 L에 누적되면 레벨 L에서의 가장 최근의 CB_h를 병합하여 새로운 레벨 (L+1)을 생성한다. 그리고 레벨 (L+1)의 끝 시간 t_e^{L+1}을 갱신한다. 따라서 n개의 스트림 데이터에 대해 계층구조가 구성되면, 스트림 환경에서 메모리로 인한 제약조건을 해결하기 위해 각 레벨에서 가장 최근의 레벨 스트림만 남겨둔다.

3.2 구조적 표현

이 절에서는 그림4의 초기 스트림구조에서 다중 스트림이 수행되는 과정을 레벨정합(Level Matching)과 누적정합(Accumulation Matching)으로 구분하여 스트림 구조가 표현되는 방법에 대해서 살펴본다.

3.2.1 레벨정합

다중 스트림 레벨에서 레벨 정합은 레벨 정합 함수를 이용하여 다중 스트림 구조를 표현하게 되며, 이 기법은 간단한 스트림 정합구조로서 멀티미디어, 이미지 처리 등을 위한 모바일 스트리밍이다. 스트리밍 계층구조 h에서 레벨 정합 함수를 이용한 스트리밍 구조는 다음과 같이 정의된다.

(정의3) 스트리밍 계층구조 h에서 레벨정합은 $LM(h)=\{\sum_{h} stream/|h|\}$이다.

예를 들어 CB_{ts}=T, CB_h=2, ts=8인 그림4에서 L→0의 스트림은 T가 도착될 때 정합이 시작되며, L에 누적된 새로운 스트림은 새로운 스트림 레벨 (L+1)로 정합 된다. 이후 가장 최근의 8개의 스트림들이 계층 구조의 각 레벨에 남게 된다. CB_{ts}=2라 할 때 그림3의 sq_1의 스트림구조를 살펴보자. L→0의 ts=2이면, 정의3에 의해 {(64+48)/2}={56}이고, ts=4이면 {(16+32)/2}={24}이다. L→1의 정합은 L→0의 두 개의 스트림을 평균하여 구한다. 즉 L→1의 ts=4이면 {(56+24)/2}={40}이 된다. 그리고 L→2의 ts=8일 때는 {(40+46)/2}={43}이 된다. 따라서 ts=2일 때는 스트림{56}과 {24}, ts=4일 때는 {40}, ts=8일 때는 {43}이 스트리밍 된다.

3.2.2 누적정합

다중 스트림 레벨에서 누적정합은 누적 정합 함수를 이용하여 다중 스트림 구조를 표현하며, 이 기법은 데이터 스트림 환경에서 메모리 제약과 스트리밍을 보다 능동적으로 수행하기 위해 사용된다. 그리고 이 기법은 서브 스트림에서 발생되는 오류를 줄이는 효과를 제공해 준다. 스트리밍 계층구조 h에서 누적 정합 함수를 이용한 스트리밍 구조는 다음과 같이 정의된다.

(정의4) 스트리밍 계층구조 h에서 누적 정합은 $AL(h)=\{\sum_{h}(stream*ts), \sum_{h} stream>$이다.

누적 정합 기법은 레벨 정합 기법과 정합 수행은 비슷하게 수행되지만 저장된 값은 다르다. 예를 들어 그림3의 sq_1에서 sq_1=64, 48, 16, 32에서 L→0의 누적 정합 스트림은 정의 4에 의해 $\{\sum(stream*ts), \sum stream\}$={1*64+2*48, 64+48}={160,112}이 된다. 그리고 16과 32는 {3*16+4*32, 16+32}={176,48}이 된다. 또한 L→1의 누적정합은 ts=4일 때 L→0의 {160, 112}, {176, 48}에 의해 생성되며, 즉 {160+176, 112+48}={336, 160}이 된다. 마찬가지로 {1144, 184}는 {616, 112}와 {528, 72}에 의해 생성되며, {1144, 184}가 된다. 그리고 L→2의 {1480, 344}는 L→1의 {336, 160}과 {1144, 184}에 의해 {336+1144, 160+184}={1480, 344}가 된다. 이와 같은 과정을 반복하여 다중 스트림 레벨 구조를 점차적으로 상위 단계로 확장하여 다중 레벨과 스트리밍을 수행하게 된다.

3.3 적합성 표현

다중 스트림이 구성되면 MT들은 sq의 세그먼트들의 적합성을 알아보기 위해 클러스터에서 최대 t개의 스트리밍 서비스를 요구할 수 있다. 이와 같은 기능을 위해서는 다중 스트림구조에서 적합한 세그먼트를 선택해야 하며, 세그먼트 적합성은 질의를 통하여 다중 스트림 계층구조에서 상위 계층과 하위계층구조로의 탐색을 수행한다. 상위 계층의 조건을 만족하는 스트림들은 정합 연산으로 인하여 탐색이 길어지며, 이때 검색된 모델은 보다 일반화된 모델이 된다. 이에 반해서 하위 계층의 조건을 만족하는 스트림은 이미 정합 연산이 수행된 스트림들이기 때문에 탐색이 용이하며, 하위계층 탐색은 상위 계층의 탐색에 비해서 명세적이다. 그러나 각 스트림 계층구조에서는 ts조건에 따라 데이터 스트림들만이 스트리밍이 수행된다. 따라서 각 ts조건을 만족하는 데이터 스트림 세그먼트에 접근하기 위해서는 적합성 여부를 수행해야 하며 적합성 여부를 수행하는 세그먼트 처리는 다음과 같이 정의된다.

(정의5) 스트림 데이터 sq_i의 세그먼트 s_j의 적합성을 수행하기 위해 j번째의 스트림 데이터를 표현하는 스트림 세그먼트 $sq_i(s_a)$는 다음과 같이 표현된다.

스트림 세그먼트 $sq_i(s_a)$에 대한 적합성은 $\{(s_a(h_1),|h_1|), (s_a(h_2),|h_2|), \ldots, ((s_ah_j),|h_j|)\}$이다. 여기서 h_j는 sq_i에서 j번째를 만족하는 스트림이다.

그리고 각 세그먼트에 대해서 적합성이 수행되면 적합성 여부에 따른 비용평가를 수행한다. 또한 sq_i에서 한 개 이상의 세그먼트가 탐색되면 전체 비용은 적합성 비용 함수를 이용하여 탐색된 세그먼트의 평균 적합성 비용을 구하며, 다음과 같이 정의한다.

(정의6) 탐색된 세그먼트의 평균 적합성 비용은 $AVR_{cost}sq_i(s_a)=\sum_{i=1}^{t}\omega_i cost sq_i(s_a)/t$이다. 여기서 t는 sq_i에서 탐색된 세그먼트의 수이고, ω_i는 세그먼트 s_i에 대한 비용 가중치이다.

평균 적합성 비용은 스트림 데이터의 성능을 알아보기 위해 사용되며, 세그먼트 sq에서 s_i일 때 스트림 데이터 두 서브스트림 sq_1(stream)과 sq_2(stream) 사이의 적합성 거리척도 $f_{distance}$는 다음과 같이 정의된다.

(정의7) 적합성 거리척도 $f_{distance}=\{\sum_{distance=1}^{n}(sq_1(stream)-$

$sq_2(stream))^2 \times |h_d|\}$이다.

예를 들어 그림3의 sq_1, sq_2, sq_3에서 n=3일 때 즉 첫 번째 대기큐가 sq_1=64, sq_2=24, sq_3=32, 두 번째 대기 큐가 sq_1=48, sq_2=38, sq_3=46일 때 적합성 거리척도 $f_{distance}$는 (정의7)에 의해 $\{(64-24)^2\times2+(24-32)^2\times2+(32-64)^2\times2\}$=5376 이고, $\{(48-32)^2\times2+(38-46)^2\times2+(46-48)^2\times2\}$=648이 된다. 따라서 적합성 거리 척도에 의한 비용은 첫 번째 대기 큐의 접근 스트림에 비해 두 번째 대기 큐의 접근 스트림 비용이 감소하게 된다.

3.4 세그먼트 관리

세그먼트 관리는 전체 스트림 데이터를 스트리밍 할 때 캐싱 블록에서 스트림 된 세그먼트와 스트림 되지 않은 세그먼트들을 관리하기 위한 기법이다. 이 기법은 자원 사용량을 최소화하여 네트워크 대역폭과 트래픽, 시간 지연과 같은 문제를 줄이기 위한 기법이다. 대기 큐에서 스트림 데이터가 자원 할당량을 최소화기 위해서는 캐싱을 위한 세그먼트 대기 큐가 준비되어야 하고 대기 큐는 순차적으로 접근된다. 이때 세그먼트 관리를 위한 각각의 세그먼트들은 구조적 표현단계를 거쳐서 패치 되며, 패치된 세그먼트들은 인코딩율과 네트워크 대역폭을 고려하여 스트리밍이 결정된다. 이처럼 스트리밍을 위한 인코딩율과 대역폭의 결정은 다음과 같이 표현된다.

$$sq(x)=\sum_{i=1}^{n} BSL_i - \frac{BSL(n+1)\times(Er-Bw)}{Bw} \qquad (1)$$

$$\frac{\sum_{i=1}^{n} BSL_i - sq(stream_i) + BSL(n+1)}{Er} \geq \frac{BSL(n+1)}{Bw} \qquad (2)$$

대역폭과 인코딩율이 결정되면, 스트리밍을 위한 버퍼링은 다음과 같이 수행된다.

$$\frac{\sum_{i=1}^{n} BSL_i - sq(stream_i)}{Er} \times B_w \qquad (3)$$

식(1)의 $sq(x)$는 임의의 세그먼트 큐이고, 식(2)(3)의 BSL_i는 i 번째의 블록 세그먼트 크기, E_r은 블록 세그먼트에 대한 평균 인코딩 율, B_w는 평균 네트워크 대역폭이다.

버퍼링이 결정되면 각 세그먼트의 효율적 관리를 위해 정적관리와 동적관리로 구분하여 관리하며, 이것은 캐싱 블록에서 스트림을 가변적으로 수행하고 관리하기 위해 사용한다.

3.4.1 정적관리

정적 관리는 sq(x)의 크기를 같은 크기로 분할하여 관리하는 기법이며, 스트림을 수행할 sq(x)에 대해서 인코딩율과 대역폭을 고려해야 하며, 정적관리 $SM(x)$는 다음과 같이 정의된다.

(정의8) $SM(x)=BSL_i(n+1)\times T_{synchro} - \frac{Er}{Bw}\times BSLi$ 이다.

여기서 $T_{synchro}$는 모바일 환경에서 정적관리를 위한 시간 동기화이다.

$SM(x)$가 결정될 때 이에 대한 버퍼 크기 $SM(x)B_{uffer}$는 다음과 같이 정의 된다.

(정의9) $SM(x)B_{uffer}=(T_{synchro}\times BSL_1)\times \frac{Er-Bw}{Er}$ 이다.

3.4.2 동적관리

동적관리는 sq(x)의 크기를 서로 다른 크기로 분할하여 관리하는 기법이며, 스트림을 수행할 sq(x)에 대해서 인코딩율과 대역폭을 고려해야 하며, 동적관리 $DM(x)$는 다음과 같이 정의된다.

(정의10) $DM(x)=BSL_i(n+1)\times T_{synchro} - \frac{Er}{Bw}\times BSLi$ 이다.

$DM(x)$가 결정될 때 이에 대한 버퍼 크기 $DM(x)B_{uffer}$는 다음과 같이 정의 된다.

(정의11) $DM(x)B_{uffer}=(T_{synchro}\times BSL_1)\times \frac{Er-Bw}{Er}$ 이다.

만일 $SM(x)$, $DM(x)=0$이면 세그먼트 관리에 대한 오버헤드가 발생된다. 이러한 문제를 피하기 $SM(x),DM(x)>0$인 $(E_r/B_w)\rfloor^{th}$ 번째의 세그먼트들에 대해서 관리를 수행하며 이 경우에는 시간지연에 따른 문제가 해결되게 된다.

4. 시뮬레이션 평가

제안된 기법의 실험 평가를 위해, 본 논문에서는 이미지 데이터와 동영상 데이터의 프레임을 세그먼트 실험 데이터로 사용한다. 시뮬레이션을 위한 세그먼트 프레임은 뉴스 비디오 프레임과 인터넷상에서 캡처한 문서 프레임을 사용하며, 실험 환경을 위해 EPM-K5500 InWireless상에서 MT에 제안된 관리 모듈을 적용하여 필드 테스트 환경을 구성하였다. 그리고 MT의 통신을 Diag Map으로 설정하고 MT의 하드웨어 키와 소프트웨어 키를 사상시켜 스트리밍을 수행하도록 하였다. 세그먼트 관리를 위한 객체 수행 비트율은 1.28Mbps, 스트리밍을 위한 파일타입은 동영상파일과 정지영상파일로 제한하였다. 스트리밍을 위한 파일 상연시간은 1분 이내로 제한하였으며, 크기는 5MB 이내로 제한하였다. 또한 각 링크 대역폭은 10/100Mbps로 하고 평균 링크 대역폭은 약 1.2Mbps로 하였으며, 제안된 기법의 적합성을 알아보기 위하여 동적 데이터 스트림 관리와 정적 데이터 스트림 관리로 스트리밍을 수행하며, 동적 데이터 스트림은 동영상 데이터 스트리밍을 위한 것이고 이 데이터는 연속적인 프레임들로 구성되어 있다. 동적 데이터 스트림의 적합성 제어를 위한 임계값은 ω이며, 데이터의 프레임 길이는 F_{length}로 나타낸다. 정적 데이터 스트림의 스트리밍 수행방법은 동적 데이터 스트림과 같으며, 연속된 프레임들 중 t시간동안 j 번째의 프레임 샘플추출은 다음과 같이 수행한다[1].

$$F_j(t)=(s+\eta)\times R_j(t)+\epsilon(t)\text{이다.} \qquad (4)$$

여기서 s는 동적 데이터 스트림과 정적 데이터 스트림의 프레임 객체이고, η와 $\epsilon(t)$는 표준 정규분포로부터 유도된 값이다. 그리고 t의 범위는 [0, length-1], $R_j(t)$는 프레임의 관계성으로서 다음과 같은 규칙에 따라 결정된다.

j 번째 프레임 객체의 임계값은 ω와 $R_j(t)$에 의해 결정되며, ω는 $0 \le \omega \le 1$이다.

만일 $\omega_i \ge 0.6$이면, $R_j(t) = |R_j(t)-1|$을 수행하고, $\omega_i < 0.5$이면, $R_j(t)=1-\frac{t \bmod Length}{Length}$을 수행하여 적합성을 결정한다.

4.1 검색 성능분석

시뮬레이션을 위해 스트림의 $ts_{current}$는 1에서 20사이로 설정하며, 동적 스트림 관리와 정적 스트림 관리를 위한 세그먼트 프레임은 $\omega \ge 0.5$으로 설정한다. 그리고 이 값은 검색시간, 응답율, 적합율을 분석하기 위해서 사용된다. 블록 캐싱의 히트에 따른 검색시간은 데이터 포인터 수와 데이터 스트림의 수를 가변적으로 변화시켜 수행한다. 블록 캐싱의 히트율이 높으면 높을수록 스트리밍 서비스율은 증가하게 되며 스트리밍 서비스율이란 클러스터링 구조에서 관련된 데이터 프레임을 스트리밍하기 위한 척도이며 서비스율(λ)는 다음과 같다.

$$\lambda = \sum_{c_i}[r_i \times (1 - \prod_{\pi,j}(1-\omega_{i,j}))]/F_{number} \text{이다.} \quad (5)$$

여기서 r_i는 클러스터의 반경이며, ω는 클러스터링 구조에서 관련된 데이터 프레임이 히트될 임계값이다. 그리고 평균 검색시간이란 질의를 수행한 후의 스트리밍이 수행될 관련된 데이터 프레임을 검색하는 시간을 말하며 다음과 같다.

$$F_{time}(t) = \frac{\sum_{c_i} F_{search} \times CB \times \lambda}{\sum_{c_i} \times F_{number}} \times \omega \quad (6)$$

가변 클러스터링의 관리의 검색결과를 알아보기 위하여 비용 함수를 이용한 원시 데이터 스트림들에 대해서 k-means 기법, C/S 서버기법, CDN 기법, 그리고 우리의 기법을 비교 분석함으로서 검색 결과를 파악하게 된다. 제안된 관리 기법에서 초기 클러스터링 구축과정은 성능 결과에 중요한 영향을 미치기 때문에 최상의 클러스터링이 구축되어 있다고 가정한다. 따라서 제안된 기법에서 검색 결과의 성능 평가를 위한 오퍼레이터는 <표1>과 같으며

표 1. 성능 평가를 위한 오퍼레이터
Table 1. Operator for performance

operator	의 미
F_{number}	클러스터링 구조에서의 프레임의 수
F_{search}	클러스터링구조에서의 스트리밍을 수행할 프레임의 수
CB	캐싱블록 크기
ω_i	임의의 세그먼트에 대한 임계값

검색 결과의 성능을 위한 캐싱블록 크기와 수를 $F_{number}=F_{search}=CB$라 하고, 10개의 클러스터링을 구성하여 $\omega \ge 0.5$로 하여 시뮬레이션을 수행하였다. 그리고 데이터 스트림의 수를 500, 1000, 1500, 2000, 2500으로 하고, ω=0.8로 하여 5회 반복 수행했을 때 검색을 수행한 내용은 다음과 같다.

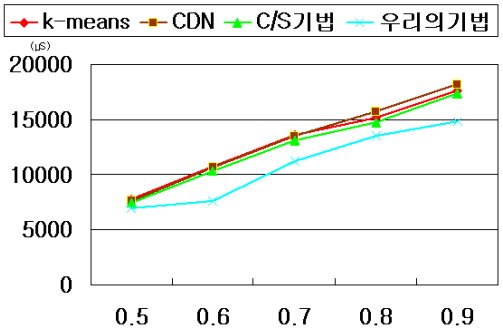

그림 5. $\omega \ge 0.5$일 때 평균 검색시간
Fig. 5. Average retreival time with $\omega \ge 0.5$

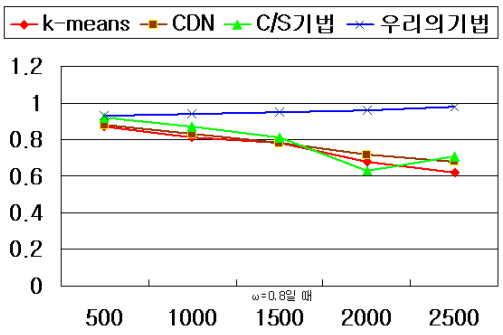

그림 6. ω=0.8일 때 평균 검색율
Fig. 6. Average retrieval rate with ω=0.8

그림에서 보듯이 평균 검색 시간은 ω이 0.5에서 0.9로 증가될수록 제안된 기법의 성능이 향상됨을 알 수 있으며, 평균 검색율 또한 향상됨을 알 수 있다. 그림6에서 ω가 0.5이하일 때는 실행시간에 크게 영향을 미치지 않기 때문에 우리의 논문에서는 편의상 ω=0.8일 때 평균 검색율을 평가하였다.

4.3 적합성 분석

이 절에서는 계층구조에서 스트림 레벨(stream), 블록캐싱 크기(CB), 세그먼트 크기에 따라 적합성을 분석한다. 일반적으로 클러스터링 관리 시간은 비용시간에 의해 결정되며, 스트림 레벨(stream) L→1에서 L→4는 클러스터링 비용과 실행시간에는 많은 영향을 미치지 않는다. 이 절에서의 적합성은 앞 절에서 논의된 평균 적합성 비용과 적합성 거리 척도에 의해서 수행되며, 적합성 분석을 위해 5개의 데이터 스트림 클러스터링 구조를 생성하여 시뮬레이션을 수행하였다.

생성된 각 데이터 스트림 집합에 대해서 50개의 동적 데이터 스트림을 클러스터에 포함하여 데이터 프레임이 생성되도록 하였다. 그림7과 그림8에서 보듯이 ω=0.8, ω=0.9로 설정하였을 때 데이터 스트림의 적합율은 비교 기법들 가운데서 비교적 성능이 우수한 C/S 기법과 비교할 때 적합성 성능이 약 10% 정도가 개선됨을 알 수 있으며, ω=0.5보다 작을 때는 관련성에 영향을 미치므로 본 실험에서는 ω=0.5 이상인 데이터 프레임들에 대해서만 스트리밍을 수행하여 평가하였다. 그리고 그림9는 LM과 CM에 대한 적합성 분석이며 적합성 비용 평가는 클러스터링을 관리 했을 때와 그렇지 않았을 때를 비교할 때 클러스터링을 관리 할 때가 적합성 비용 측면에서 더 효율적임을 알 수 있다.

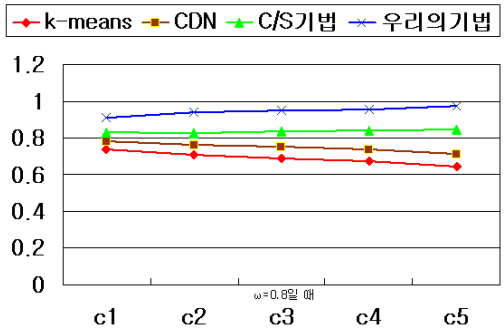

그림 7. ω=0.8일 때의 적합율
Fig. 7. Fitness rate with ω=0.8

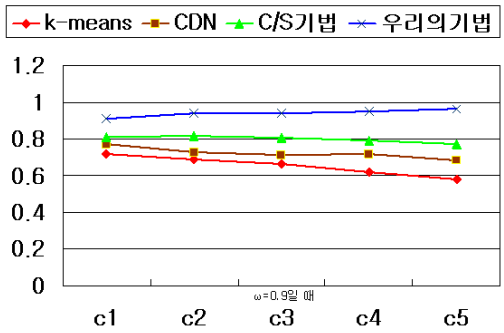

그림 8. ω=0.9일 때의 적합율
Fig. 8. Fitness rate with ω=0.9

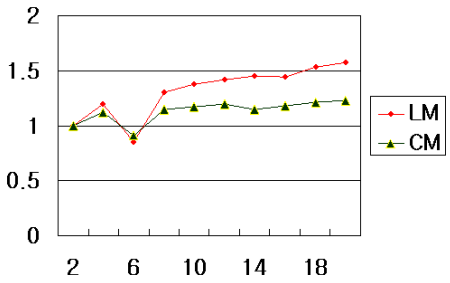

그림 9. LM과 CM의 비용 적합성 분석
Fig. 9. Analysis of cost fitness for LM and CM

5. 결 론

최근에 분산 모바일 환경에서 동적 데이터 스트림과 정적 데이터 스트림을 위한 여러 기법들이 제안되고 있다. 데이터 스트림이 구조화 되었을 때 스트림 관리는 모바일 성능과 QoS에 중요한 영향을 미치게 되며, 본 논문에서는 이러한 성능 관리를 위해 효율적인 가변 클러스터링 관리기법을 제안하였다. 제안된 기법은 클러스터링 환경분석을 통하여 스트림들을 분석하게 되며 구조적 표현단계를 통하여 클러스터링의 구조와 스트림 구조를 파악하게 된다. 구조적 표현단계에서 다중 스트림이 수행되는 과정은 레벨정합과 누적정합으로 구분하여 스트림 구조를 표현하였으며, 적합성 표현단계에서는 스트림의 적절성 파악을 위하여 평균 적합성비용과 적합성 거리 척도를 제안하였다. 그리고 스트림 데이터를 스트리밍할 때 블록 캐싱이 수행될 수 있도록 인코딩율과 대역폭에 따라서 스트림이 가변적으로 관리되도록 동적관리와 정적관리로 구분하여 스트림이 수행되도록 하였다. 이러한 제안된 기법의 성능 평가를 위해 검색 성능과 적합성에 따라 효율성을 분석하였다. 분석결과 우리의 기법이 k-means 기법, C/S 서버기법 그리고 CDN 기법에 비해서 검색시간, 검색율 및 적합성이 우수함을 알 수 있다.

참 고 문 헌

[1] Bi-Ru Dai, Jen-Wei Huang, Mi-Yen, and Ming-Syan Chen, "Adaptive Clustering for Multiple Evolving Streams," *IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING*, VOL. 18, NO. 9, PP. 1166-1180, 2006.

[2] A. Bulut and A. K. Singh, "Swat: Hierarchical Stream Summerization in Large Network," *Proc. Int'l Conf.* pp. 303-314. 2003.

[3] Mi-Yen Yen, Bi-Ru Dai, and Ming-Syan Chen,"Clustering over Multiple Evolving Streams by Events and Correlations," *IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING*, VOL. 19, NO. 10, PP. 1349-1361, 2007.

[4] T. Johnson, S. Muthukrishnan, and I. Rozenbaum," Sampling Algorithm in a Stream Operator," *Proc. ACM SIGMOD conf.*, pp. 1-12, 2005.

[5] S. Guha, N. Mishra, R. Motwani and L. O'Callaghan, "Clustering Data Streams," *Proc Symp. Foundation of Computer Science*, pp. 359-366, 2000.

[6] L. O'Callaghan, N. Mishra, A. Meyerson, S. Guha, and R. Motwani, " Streaming-Data Algorithms for High-Quality Clustering," *Proc. Int'l Conf. Data Eng*, 2002.

[7] C. C Aggarwal, J. Han, J. Wang, and P.S. Yu," On Demand Classification of Data Streams," *Proc. ACM SIGKDD*, pp. 503-508, 2004.

[8] Charu C. Aggarwal, Jiawei Han, Jianyong Wang, and Phillips S. Yu,"A Framework for Demand Classification of Evolving Data Streams," *IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING*, VOL. 18, NO. 5, PP. 577-589, 2006.

[9] Songqing Chen, Bo Shen, Susie Wee, and Xiaodong Zhang, " Segment-Based Streaming Media Proxy: Modeling and Optimization," *IEEE TRANSACTION on MULTIMEDIA*, VOL. 8, NO. 2, pp. 243-256, 2006

[10] J. Jung, B. Krishnamurthy, and M. Rabinovich, "Flash Crowds and Denial of Service Attacks: Characterization and Implications for CDN's and Web sites," *in Proc*, Int'l www Conf. 2002.

[11] Ewa Kusmierrek, Yingfei Dong, and David H. C. Du,"Lookback: Exploiting Collaborative Caches for

Large Scale Streaming," *IEEE TRANSACTIONS ON MULTIMEDIA,* VOL. 8, NO. 2, pp. 233-242, 2006.
[12] C. Aggarwal, J. Wolf, and P. Yu,"On optimal batching policies for Video on Demand Storage Servers," *in Proc. Int'l Conf.* Multimedia Systems'96, 1996.
[13] Zhe Xiang, Qian Zhang, Wenwu Zhu, Zhensheng, and Yu-Qin Zhang,"Peer-to-Peer Based Multimedia Distribution Services," *IEEE TRANSACTIONS ON MULTIMEDIA*, VOL. 6, NO. 2, pp. 343-354, 2004.

저 자 소 개

정택원(Taegwon Jeong)
1981년 2월 : 서울대학교 대학원전기공학과 (공학석사)
1991년 : Univ, of Florida Dept of EE (Ph.D)
1983년～1998년 : ETRI 책임연구원
1998년～현재 : 전북대학교 응용시스템 공학부 교수
관심분야 : 정보통신, 이동통신
E-mail : ttwjeong@yahoo.co.kr

이종득(Chong Deuk Lee)
1989년 : 전북대학교 대학원 전산통계학과 (이학석사)
1998년 : 전북대학교 대학원 전산통계학과 (이학박사)
1992년～2002년 : 서남대학교 컴퓨터정보통신학과 교수
2002년～현재 : 전북대학교 응용시스템 공학부 교수
관심분야 : Mobile Ad-hoc Networks, Sensor Networks, 멀티미디어 통신, Mobile 성능평가
E-mail : cdlee1008@chonbuk.ac.kr