온라인 멀티미디어 전처리를 위한 분산 객체 활성화

허진경, 이웅기
조선대학교 전산통계학과

On-Line Multimedia Pre-Processing for Distributed Object Activation

Jin-Kyoung Heo, Woong-Ki Lee
Dept. of Computer Science & Statistics, Chosun University
Email : heojk@shinbiro.com, wglee@chosun.ac.kr

요 약

클라이언트로부터 전송되는 멀티미디어 데이터를 실시간으로 서버에서 분석하여 처리하는 시스템의 경우에는 입력되는 데이터의 양에 비례하여 서버의 부담은 증가하게 된다. 이때 서버에서는 데이터의 병목현상이 발생하게 되고, 이는 바로 전체적인 시스템의 성능을 저하시키는 결과를 초래하게 된다. 본 논문에서는 데이터 처리의 병목현상을 해결하여 시스템의 성능을 높일 뿐만 아니라, 인터넷상의 유휴 서버들을 활용할 수 있게 하기 위한 방안으로 분산처리 기술을 이용한 전처리 작업 시스템과 그 성능을 향상시키기 위한 분산 객체 활성화 시스템을 제안한다.

1. 서론

네트워크 상에서, 클라이언트에서 만들어진 데이터를 서버로 실시간으로 전송하고, 이를 또한 서버에서 실시간으로 분석 및 처리하는 시스템은 하나의 서버에 여러 대의 클라이언트를 갖게 된다. 클라이언트에서 하나의 데이터가 서버에 입력되더라도 서버에서는 입력되는 데이터를 사용하기 위해서는 많은 전처리 작업들을 거치게 되는데, 클라이언트의 수가 많을수록 서버에 입력되는 데이터의 양 또한 증가하게 되며, 데이터의 증가는 그 증가되는 양에 비례하여 서버에 많은 부담을 주게 된다. 지금까지는 이러한 데이터들을 처리하는데 하나의 서버가 전처리와 후처리를 담당하게 하는 방법을 사용하였다. 즉, 클라이언트로부터 데이터가 발생하면 이의 모든 처리를 하나의 서버에서 처리하는 방법을 사용하고 있었다. 특히 멀티미디어 데이터의 경우에는 클라이언트로부터 발생된 데이터를 서버에서 최종 처리에 이르기까지의 단계 중에서 데이터의 전처리 작업에 소요되는 단계가 많은 부분을 담당한다.[1][4][5][6]

온라인상에서 사용자의 지문, 홍채, 얼굴 그리고, 음성 등을 통하여 사용자를 검증하기 위해 특징점 추출, 분류, 매칭 들을 통한 여러 가지 방법들이 연구되고 있고, 이를 위한 효과적인 시스템의 필요성이 대두되고 있는 만큼 검증단계 이전의 영상의 전처리 작업에 관한 온라인 처리를 하기 위한 방법으로, 네트워크 상에서 멀티미디어 데이터의 온라인 실시간 전처리 시스템을 제안하고자 한다.

본 논문의 구성은 다음과 같다. 먼저 2장에서 네트워크 프로그래밍 아키텍처에 해당하는 Tiered Model에 대하여 기술하고, 3장에서는 분산 처리기술에 필터링을 적용한 개념에 대하여 기술하고, 이를 이용한 사용자 멀티미디어 데이터 전처리 시스템에 적용한다. 그리고, 4장에서는 성능의 향상과 시스템의 중단 현상을 최소화 하기 위한 방안으로 객체 활성화를 통한 멀티미디어 데이터 전처리 시스템을 제안하고 있다. 마지막으로, 5장 결론에서는 제안한 시스템을 시뮬레이션하여 그 유효성을 보이고, 향후의 연구 과제를 제시한다.

2. Tiered 모델

데이터베이스를 사용한 네트워크 프로그래밍 모델을 생각하면 먼저 단순히 클라이언트 자체가 서버의 역할을 하는 1 Tier 모델을 생각할 수 있다. 다음 그

림은 1 Tier모델을 나타내고 있다.

그림 1. One Tiered Model

위와 같은 1 Tier 모델의 경우에는 데이터베이스가 있는 컴퓨터를 직접 사용자가 조작하여야 하기 때문에, 데이터 조작에 능숙한 사용자가 필요하고, 네트워크 환경에 부적합하다는 단점이 있다.

다음은 서버와 클라이언트로 분리하는 2 Tier 모델을 생각하게 된다. 2 Tier 모델의 경우에도 데이터는 데이터베이스 서버에 존재하게 되고, 사용자의 데이터를 입력받아 처리하는 시스템은 데이터베이스 프론트엔드에 존재하게 되는 모델을 생각할 수 있다. 다음 그림은 2 Tier 아키텍처 모델을 나타내는 그림이다.

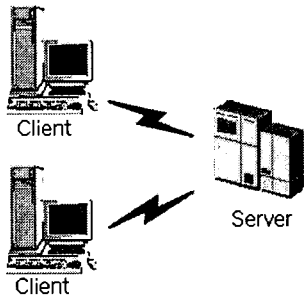

그림 2. Two Tiered Model

위의 그림은 인터넷이 확산되기 이전의 초창기 모델이다. 주로 인트라넷 프로그램들이 이러 했으며, 위의 모델은 데이터베이스 서버가 있고 사용자들이 사용하는 워크스테이션에 데이터베이스에 접근하는 코드를 포함하는 방식이다. 이러한 방식의 프로그래밍 아키텍처의 경우에는 많은 단점을 포함하게 된다. 가장 대표적인 예로서 서버의 주소 또는 포트 번호가 변경되는 등의 사소한 경우뿐만 아니라, 데이터베이스를 새로운 것으로 바꿀 경우에도 모든 사용자의 워크스테이션에 설치되어 있는 프로그램을 갱신하여야 할 것이다. 요즘 같은 글로벌 시대에는 상당한 비용이 소요될 뿐만 아니라, Mirroring, Caching, Proxy services, Secure Transaction등을 해결하는 데 있어서 많은 문제점들을 내포하게 된다.[1][3]

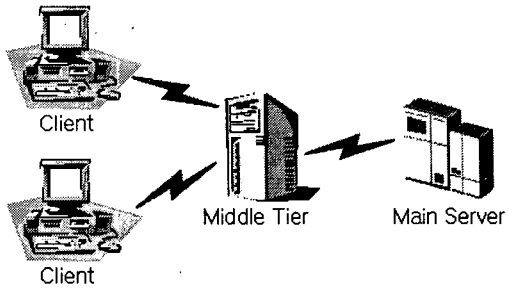

그림 3. Three Tiered Model

2 Tier 모델 이후에 제안된 모델이 3 Tier 모델이다. 3 Tier 어플리케이션이란 3개의 주요 부분으로 구성되어 있는 응용프로그램으로서, 각각은 네트웍 상의 서로 다른 장소에 분산되어 있다. 여기서 3개의 주요 부분이란 Presentation Tier, Business Logic tier, Data Tier로 나누어진다. 3 Tier 어플리케이션에서, Presentation Tier에 해당하는 프로그램 사용자의 워크스테이션은 GUI를 제공하는 프로그램과, 특정 프로그램에 맞는 입력 양식이나 인터랙티브 윈도우 등을 가지고 있다. Business Logic Tier는 네트워크상의 다른 공유된 컴퓨터상에 위치하면서, 사용자 워크스테이션으로부터의 클라이언트 요청에 대해 마치 서버처럼 행동한다. Business Logic Tier는 차례로 어떤 데이터가 필요한지를 결정하고, 메인프레임 컴퓨터 상에 위치하고 있을 세 번째 계층의 프로그램에 대해서는 마치 클라이언트처럼 행동한다. 세 번째 계층에 해당하는 Data Tier는 데이터베이스와 그것에 액세스해서 읽거나 쓰는 것을 관리하는 프로그램을 포함한다. 어플리케이션의 구조는 이보다 더 복잡해질 수 있지만, 3 Tier 관점은 대규모 프로그램에서 일부분에 관해 생각하기에 편리한 방법이다.[1][2]

3 Tier의 장점은 어플리케이션이 클라이언트/서버 컴퓨팅 모델을 사용하기 때문에, 3 Tier에서 각 부분은 각기 다른 팀의 프로그래머들에 의해 각기 다른 언어를 사용하여 동시에 개발될 수 있다. 어떤 한 계층의 프로그램은 다른 계층에 영향을 주지 않고도 변경되거나 위치가 달라질 수 있기 때문에, 3 Tier 모델은 새로운 요구나 기회가 생길 때마다 어플리케이션을 지속적으로 변화시켜야하는 기업이나 소프트웨어 패키지 개발자들이 이에 쉽게 대처할 수 있게 해준다. 기존의 애플리케이션들은 영구적으로 또는 일시적으로 계속 유지될 수 있으며, 하나의 컴포넌트로서 새로운 계층 내에 캡슐화 될 수도 있다. 실제 많은 사용자 인증 처리들이 3 Tier 모델을 기반으로 구축되어 있

기도 하다.[2][4][6]

3 Tier 모델을 기반으로 한 멀티미디어 데이터의 처리 방법은 다음과 같다.

(1) 클라이언트에서 발생한 데이터는 여과 없이 바로 서버에 보내진다.
(2) 서버는 각 클라이언트의 데이터를 입력받아 처리에 적합한 데이터로 만들기 위해 다시 전처리 작업을 한다.
(3) 전처리 작업이 끝난 데이터를 이용하여 서버에서는 최종 처리 절차에 들어간다.
(4) 서버의 처리 결과를 클라이언트에 전달한다.

하지만 위와 같은 방법에 있어서는 사용자의 수가 증가될 경우 또는 데이터 량이 클 경우에는 인증 처리에 소요되는 시간보다 전처리 작업에 더 많은 시간을 할당할 수도 있게 된다. 또한 이로 인한 데이터의 병목현상을 초래할 수도 있다.

3. 분산 필터링

3 Tier 어플리케이션 아키텍처는 분산 객체지향 프로그래밍과 사상이 일치한다.

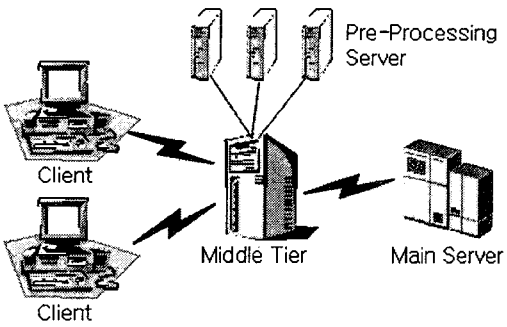

그림 4. Distributed Filtering

위의 그림은 분산 필터링 시스템의 구조를 나타낸 것이다. 클라이언트는 사용자의 화면을 중앙의 서버에 보내게 되는데, 이때 사용자의 인증에 필요한 영상의 전송은 객체 직렬화를 통하여 이루어지게 함으로써 전송 비용을 최소화하게 된다. 중앙의 서버는 각 사용자들의 데이터를 입력받아 바로 인증 처리에 들어가는 것이 아니고, 클라이언트의 데이터를 인증 처리에 적합한 데이터로 만들기 위해 다시 전처리 작업에 들어가게 된다. 이때 전처리 작업을 중앙의 서버에서 수행되는 것이 아니고, 전처리를 전담하는 보조 서버에 담당하게 하는 방법이다. 전처리만을 담당하는 전용 보조 서버의 메모리에는 그 시스템에서 최적으로 수행될 수 있는 처리 작업이 적재되어 있어 특정 전처리 작업에 최상의 성능을 발휘할 수 있게 된다. 이처럼 분산 처리는 한곳에 집중되는 작업을 분산시켜 동시에 처리하게 함으로써 보다 빠른 결과를 얻기도 하지만, 인터넷상의 유휴자원을 사용할 수 있다는 측면에서도 많은 각광을 받고 있다.

4. 분산 객체 활성화

분산처리 기술은 많은 장점을 가지고 있지만, 특정 보조 서버에 하나의 전처리 작업을 위임한다는 것은 아주 큰 문제점을 포함하기도 한다. 만일 전처리 작업을 하는 보조서버 중에서 잡음을 제거하는 전담 서버에 시스템 오류가 발생했다고 가정하자. 이 경우에는 하나의 보조서버의 오류로 인해 전체 시스템의 마비를 가져오게 된다.

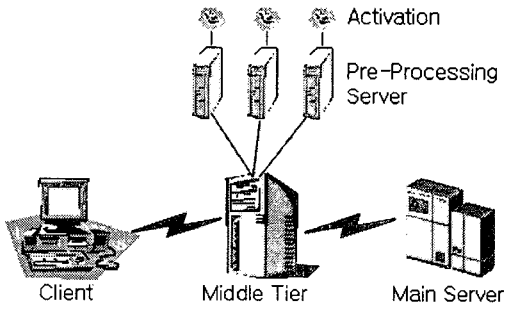

그림 5. Object Activation

하나의 보조 서버의 중단으로 인해 전체 시스템 마지를 방지하기 위해서 동일한 보조서버를 둔다는 것은 자원관리에 큰 허점으로 보인다. 이를 위해 그림 5에서는 특정 전처리를 담당하는 보조 서버들에 하나의 보조기억장치를 두고 있는데, 이는 단순한 데이터를 백업 받기 위한 하드디스크를 의미하지는 않는다. 그림이 의미하고 있는바가 바로 Object Activation을 의미하는데, 이는 분산처리에 있어서 아주 가끔씩 호출되는 객체가 있을 경우에 이를 메모리에 올려놓는 것이 아니고, 하드디스크에 저장해 놓았다가 필요하면 가져다 쓰는 방식이다.[2] 즉, 하나의 전처리를 전담하는 보조 서버가 오류로 인해 중단되었을 경우를 대비하여 다른 전처리 보조 서버들은 자신의 전처리에 필요한 프로세스 및 객체들은 메모리에 올려놓고 처리하는 반면에 자신과 관련되지 않는 전처리 객체들은 하드디스크에 저장해 놓았다가 대비하게 하는 방식이

다. 실제 특정 보조 서버가 중단될 경우에 다른 보조 서버의 Object Activation에 의해 처리됨으로서 전체 시스템이 중단되는 것을 방지할 수 있다.

5. 결론

본 논문에서는 필터링 시스템의 분산처리를 제안하였다. 사용할 수 있는 분산처리 기술들에 대하여 많은 방법들과 이로 인해 발생할 수 있는 문제점들에 대하여 다루었다. 실험을 위해서 Pentium III 800, Ram 320 PC를 중앙의 서버로 두었고, 보다 낮은 사양의 컴퓨터(Pentium 200, Ram 128) 3대를 보조 서버로 사용하였으며, 언어는 자바, JDK 1.3.1 그리고 RMI기술을 사용하였다. 사용된 영상은 5% Salt & Pepper 잡음을 포함한 256x256 크기의 임의의 영상 100개를 사용하였다. 먼저 다음 그림들에서 실험 결과 영상을 보이고 있다.

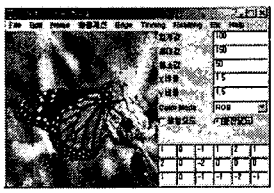

그림 6. Source Image

그림 7. 잡음감소(median)

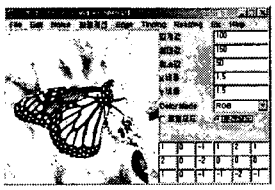

그림 8. 화면개선(Contrast)

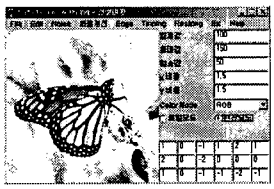

그림 9. 화면개선(선형변환)

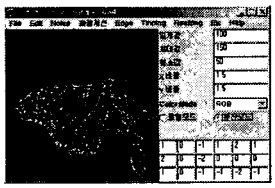

그림 10. 윤곽선추출(Sovel)

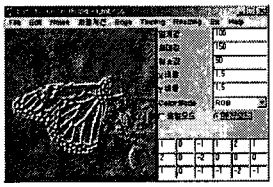

그림 11. Embossing

다음 표에서는 3 Tier Model과 분산 3 Tier 모델에서의 성능을 비교한 표이다. 5대의 클라이언트에서 동시에 데이터를 전송하여 서버에서의 처리를 요구하였을 때 각 처리되는 평균 시간을 나타낸 것이다.

Filter	3 Tier	분산 3Tier	비고
Median	680ms	520ms	잡음제거(3x3)
Contrast	95ms	84ms	화면 개선
선형변환	75ms	68ms	화면 개선
Sovel	160ms	133ms	경계선 추출
Emboss	80ms	73ms	Embossing 효과

표 1. 2Tier 와 3Tier 성능 비교

다음은 분산 3Tier 모델에서 객체 활성화를 사용했을 때와 그렇지 않았을 때의 오류상황에 대한 비교를 나타낸 것이다. 오류상황은 응답시간을 기준으로 자바에서 예외 처리를 사용하였다.

객체 활성화	사용	사용 안함	비고
오류 발생	0%	3.8%	
처리 성능	70.8%	처리 불가	

표 2. 오류 처리 성능

논문에서 제안된 방법은 서버의 작업을 분산시킴으로 인해 서버의 부담으로 인한 속도의 저하와 이로 인한 비용의 증가문제들을 해결할 수 있다. 향후 과제로는 시스템의 최적화를 통하여 보다 낮은 결과를 유도함과 동시에, 향 후 시스템의 유지보수 측면을 위해 한번 구축한 시스템을 이후에도 변경 없이 사용함으로써 또 다른 필터링 알고리즘의 추가로 인한 전체 시스템의 업그레이드 문제를 해결하기 위한 방안에 대하여 연구하고자 한다.

[참고문헌]

[1] Advanced Java programming with Workshop, Sun Microsystems, 2000

[2] Distributed Programming With Java Technology, Sun Microsystems, 2001

[3] Sun Microsystems, JDK 1.3 Documentation <http://java.sun.com/>

[4] Andreas Vogel and Keith Duddy, 'Java Programming with CORBA', John Wiley & Sons, 1997

[5] Marc H. Brown and Marc A. Najork , "Distributed Active Objects", Computer Networks and ISDN Systems, Vol. 28

[6] Robert Orfali and Dan Harkey, "Client/Server Programming with Java and CORBA", John Wiley & Sons, 1997