논문 2011-1-14

객체 활성화를 이용한 분산 영상처리

Distributed Image Preprocessing using Object Activation

허진경*

Jin-Kyoung Heo

요 약 영상처리시 처리 요구 되는 영상데이터의 크기에 비례하여 시스템의 부담은 더욱 더 증가하게 된다. 특히 클라이언트로부터 전송되는 영상 데이터를 실시간으로 서버에서 분석하여 처리하는 원격 영상처리 시스템에서는 처리해야 할 데이터가 집중되는 서버 측에는 데이터의 병목현상이 발생하게 되고, 데이터의 병목현상은 시스템의 성능을 저하시킬 수 있다. 이를 해결하기 위해 여러 대의 서버를 두어 처리하는 단순 분산처리를 사용하여 해결 할 수 있으나, 이는 그만큼의 비용 부담을 가져올 수 있다. 본 논문에서는 영상 데이터 처리의 병목현상을 해결하기 위한 분산 시스템에 객체 활성화 기술을 접목시킨 새로운 형태의 분산 영상 처리를 제안하고자 한다. 제안한 방법은 영상 전처리 작업을 최적화 된 시스템에 의해 처리하게 하여 더 높은 성능을 가져올 수 있다. 또한 최적화된 서버의 오류로 인해 정상적으로 동작하지 않을 경우를 대비하여, 네트워크 내의 유휴서버를 활용하여 서비스를 제공할 수 있다.

Abstract Server overload is directly proportional to requested image data size in a image processing. If request data are increase then system is overloaded in a image processing system. For the reduce of server bottle neck, we will be able to consider a distributed processing. Simple distributed processing system can solve server bottleneck and system overload but high cost system requirements. In this paper, Proposes a new distributed image processing system. Object activation technology are being grafted on to simple distributed processing system. It can optimize the user of system resources and can reuse idle system resources in network.

Key Words : Object Activation, Remote Preprocessing, Image Processing.

Ⅰ. 서 론

네트워크상에서, 클라이언트에서 만들어진 데이터를 서버에 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[1]. 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.

온라인상에서 사용자의 지문, 홍체, 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 원격 영상데이터의 분산처리 시스템을 제안하고자 한다.

*정회원, 호원대학교 사이버수사경찰학부
접수일자: 2011.1.3 수정일자: 2011.2.1
게재확정일자: 2011.2.11

본 논문의 구성은 다음과 같다. 먼저 2장에서 네트워크 프로그래밍 아키텍처에 해당하는 분산처리와 미들웨어 시스템에 대하여 기술하고, 3장에서는 객체 활성화와 분산처리 개념에 대하여 기술하고, 성능 향상과 시스템의 중단 현상을 최소화하기 위한 방안으로 객체 활성화를 통한 영상처리 시스템을 제안하고 있다. 4장에서는 제안한 시스템을 시뮬레이션 하여 그 유효성을 보이고, 5장에서는 향후의 연구 과제를 제시한다.

II. 분산처리와 미들웨어 시스템

데이터베이스를 사용한 네트워크 프로그래밍 모델을 생각하면 서버와 클라이언트로 분리하는 2 Tier 모델을 생각하게 된다. 2 Tier 모델의 경우에도 데이터는 데이터베이스 서버에 존재하게 되고, 사용자의 데이터를 입력받아 처리하는 시스템은 데이터베이스 프론트 엔드에 존재하게 되는 모델을 생각할 수 있다[3]. 2 Tier 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[7].

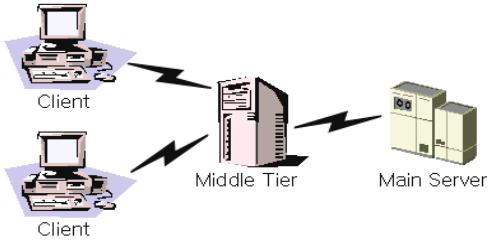

그림 1. 3티어 모델
Fig. 1. Three Tiered Model

2 Tier 모델 이후에 제안된 모델이 3 Tier 모델이다. 3 Tier 어플리케이션이란 3개의 주요 부분으로 구성되어 있는 응용프로그램으로서, 각각은 네트워크 상의 서로 다른 장소에 분산되어 있다. 여기서 3개의 주요부분이란 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[2,3].

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[5,6]. 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 있기도 하다.

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.

Ⅲ. 객체 활성화와 원격 영상처리

1. 분산 필터링

3 Tier 어플리케이션 아키텍처는 분산 객체지향 프로그래밍과 사상이 일치한다.

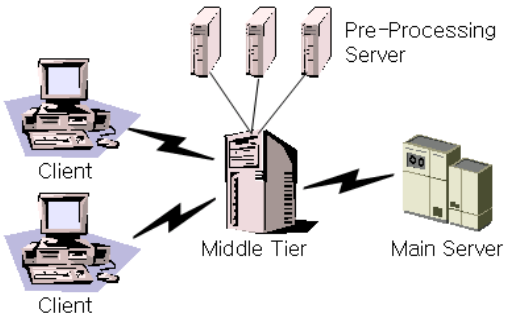

그림 2. 분산 필터링
Fig. 2. Distributed Filtering

위의 그림 2는 분산 필터링 시스템의 구조를 나타낸 것이다. 클라이언트는 처리할 영상 데이터를 중앙의 서버에 보내게 되는데, 이때 데이터의 전송은 객체 직렬화를 통하여 이루어지게 함으로써 전송 비용을 최소화하게 된다[7]. 중앙의 서버는 각 사용자들의 데이터를 입력받아 바로 영상처리에 들어가는 것이 아니고, 클라이언트의 데이터를 영상처리에 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.

2. 객체 활성화를 이용한 원격 영상처리 시스템

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 보조 서버의 오류로 인해 전체 시스템의 마비를 가져오게 된다.

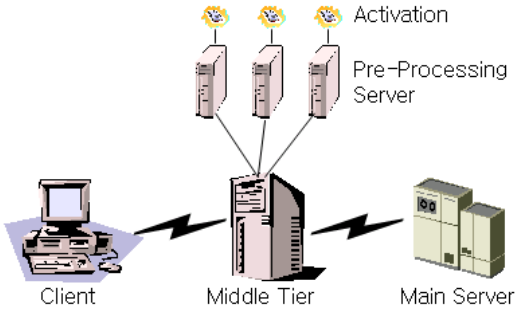

그림 3. 객체 활성화를 이용한 분산 영상처리
Fig. 3. Distributed image processing using object activation

하나의 보조 서버의 중단으로 인해 전체 시스템 마지를 방지하기 위해서 동일한 보조서버를 둔다는 것은 자원관리에 큰 허점으로 보인다. 이를 위해 그림 3에서는 특정 전처리를 담당하는 보조 서버들에 하나의 보조기억장치를 두고 있는데, 이는 단순한 데이터를 백업 받기 위한 하드디스크를 의미하지는 않는다. 그림이 의미하고 있는바가 바로 객체 활성화 시스템을 의미하는데, 이는 분산처리에 있어서 아주 가끔씩 호출되는 객체가 있을 경우에 이를 메모리에 올려놓는 것이 아니고, 하드디스크에 저장해 놓았다가 필요하면 가져다 쓰는 방식이다. 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 방식이다. 실제 특정 보조 서버가 중단될 경우에 다른 보조 서버의 객체 활성화 시스템에 의해 처리됨으로서 전체 시스템이 중단되는 것을 방지할 수 있다.

Ⅳ. 분산 영상처리 실험

본 논문에서는 화상 인식 시스템에서 주가 되는 필터링 시스템의 분산처리를 제안하였다. 사용할 수 있는 분산처리 기술들에 대하여 많은 방법들과 이로 인해 발생할 수 있는 문제점들에 대하여 다루었다. 실험을 위해서 Intel Core2 Quad CPU 2.40GHz, Ram 2GB를 메인 서버로 하였다. 언어는 JAVA(JDK 6.0), 그리고 RMI기술을 사용하였다. 사용된 256x256 크기의 Lena 영상을 사용하였다. 잡음제거에 사용된 영상은 원 영상에 5% 가우시안 잡음을 삽입한 영상을 사용하였다. 다음 그림들에서는 실험 데이터의 전처리 결과를 보여주고 있다.

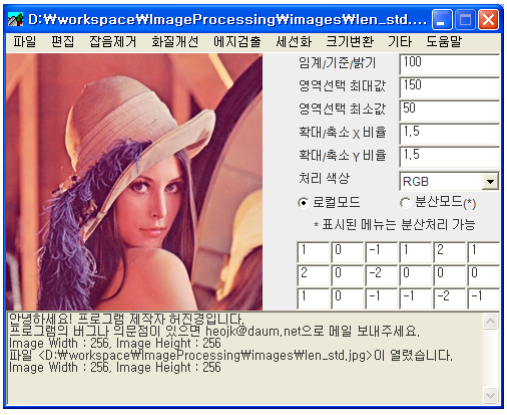

그림 4. 원 영상
Fig. 4. Original image

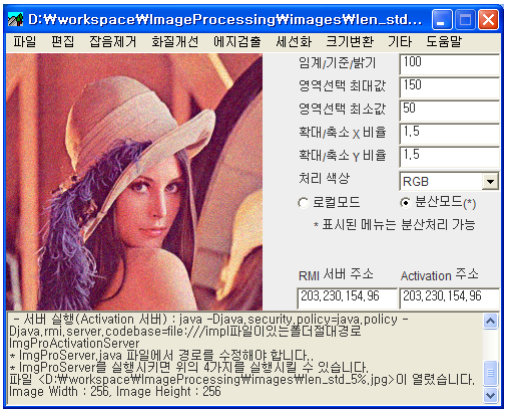

그림 5. 5% 가우시안 잡음 영상
Fig. 5. 5% Gaussian noise image

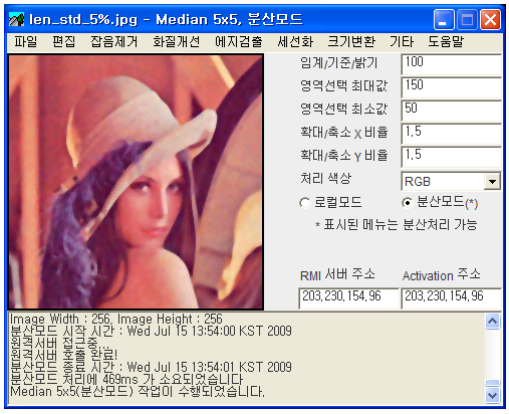

그림 6. 잡음제거(메디안)
Fig. 6. Noise reduction(Median)

그림 7. 화질개선(히스토그램 균등화)
Fig. 7. Improvement of image quality (Histogram equalization)

그림 8. 윤곽선 추출(소벨)
Fig. 8. Contour extraction(Sovel)

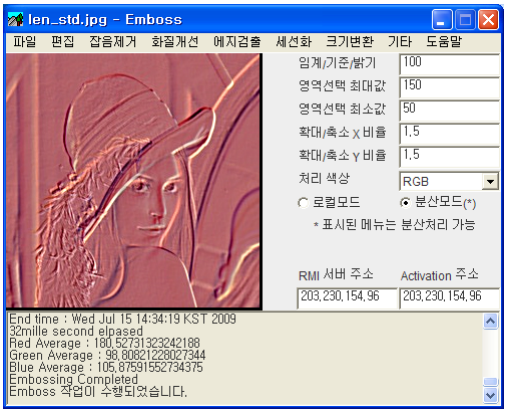

그림 9. 엠보싱 효과
Fig. 9. Embossing effect

표 1 에서는 분산 3 Tier 모델에서 처리 사간과 객체활성화를 사용한 모델에서의 성능을 비교하였다. 클라이언트에서 데이터를 전송하여 전처리 작업을 요구하였을 때 각 처리되는 시간을 나타낸 것이다. 객체 활성화를 이용한 전처리 서버에서의 최초 응답시간은 객체활성화를 사용하지 않은 서버에 비하여 전처리 작업들의 응답시간이 더 많이 요구되었다. 그러나 1회 응답 이후 활성화 서버를 사용한 전처리 응답시간은 객체활성화를 사용하였을 때와 그렇지 않았을 때의 처리시간은 큰 차이를 보이지 않고 있다.

표 1. 분산 3Tier와 객체 활성화를 이용한 전처리 성능 비교
Table 1. Performance comparison distributed 3-Tier and object activation preprocessing

처리	분산 3 Tier	객체활성화 이용		비고
		활성화 전	활성화 후	
잡음제거	469ms	734ms	453ms	Median 5x5
화질개선	1172ms	1578ms	1125ms	히스토그램 평탄화
윤곽선 추출	63ms	516ms	47ms	Sovel
효과	15ms	31ms	16ms	Embossing 효과

전처리 서버들 중에서 특정 서버의 시스템 오류는 전체 서비스에 매우 큰 영향을 줄 수 있다. 예를 들어 잡음제거를 담당하는 전처리 서버가 시스템오류로 중지되었을 경우에는 화질개선 및 윤곽선 추출 작업에 매우 큰 영향을 줄 수 있다. 그러나 객체활성화 시스템을 사용하였을 경우에는 특정 시스템의 오류 상황에도 다른 객체활성화 시스템을 통해서 지속적인 서비스를 제공할 수 있다.

표 2는 객체활성화 시스템의 처리객체 할당을 보여주고 있다. 각 서버 1부터 3까지 숫자로 나타내었으며 고유의 작업을 담당하도록 되어 있다. 전처리 작업은 A부터 C까지 알파벳으로 표기하였다. A는 잡음제거 작업을 의미하고, B는 화질개선, C는 윤곽선 추출 작업을 의미한다.

표 2. 활성화 시스템에서의 객체 할당
Table 2. Object assign of activation system

처리 서버	전처리 작업	활성화 객체
서버 1	잡음제거	B, C
서버 2	화질개선	A, C
서버 2	윤곽선 추출	A, B

표 3은 분산 3Tier 모델에서 객체 활성화를 사용했을 때와 그렇지 않았을 때의 오류상황에 대한 비교를 나타낸 것이다. 실험을 위해 256x256 크기의 5% 가우시안 잡음 영상 100개를 Median 필터링을 이용한 잡음제거 작업을 객체활성화에 사용하였다. 잡음제거를 하기위한 전처리 서버의 에러율은 응답시간을 기준으로 1% 랜덤 값으로 지정하였으며 자바에서 예외 처리를 사용하였다. 객체활성화를 사용한 시스템에서는 잡음제거 서버의 중단시에도 다른 서비스를 제공하는 활성화 서버를 통해 지속적인 서비스를 제공할 수 있었다.

표 3. 전처리 서버 오류시 시스템 대체 작동 성능
Table 3. Failover performance of object activation system

	객체 활성화 사용	객체 활성화 사용 안함	비고
처리 성능	63.9%	처리 불가	

V. 결 론

논문에서 제안된 방법은 서버의 작업을 분산시킴으로 인해 서버의 부담으로 인한 속도의 저하와 이로 인한 비용의 증가문제들을 해결할 수 있다. 네트워크 트레픽 또

는 시스템 부하를 분산시키는 로드벨런싱과는 다른 의미이다. 또한 특정 작업을 진행하는 전처리 서버의 오류시에도 객체 활성화 서버를 이용하여 지속적인 서비스를 진행시킬 수 있다. 향후 과제로는 활성화 시스템의 메모리 사용 문제와 시스템의 최적화를 통하여 보다 낳은 결과를 유도함과 동시에 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.

참 고 문 헌

[1] Alan Bateman, Michael McMahon, Networking with Java(TM) 2 Platform, Standard Edition (J2SETM): Present and Future, Sun Microsystems, Inc. Sun's Worldwide Java Developer conference, 2002.

[2] Alt, M. Gorlatch, S., "Adapting Java RMI for grid computing", Future generations computer systems, Vol.21 No.5, 2005, pp.699-707

[3] Eberhard, J. Tripathi, A., "Mechanisms for object caching in distributed applications using Java RMI", Software: Practice and Experience, Vol.37 No.8, 2007, pp.799-831.

[4] Jeff Nisewanger, Sean Mullan, Rosanna Lee, Java 2 Platform, Standard Edition(J2SE) Security Looks Ahead: New Features From Cryptography to XML Security, Java Security Engineers, Sun Microsystems, Inc. Sun's Worldwide Java Developer conference, 2003.

[5] Marques, P., "BUILDING SECURE JAVA RMI SERVERS", Dr. Dobb's journal, Vol.27 No.11, 2002. pp.36-45.

[6] Metkowski,R. Bala, P., "Parallel Computing in Java: Looking for the Most Effective RMI Implementation for Clusters", LECTURE NOTES IN COMPUTER SCIENCE, No.3911. 2006, pp.272-277.

[7] W. Hohberg, "How to find biconnected components in distributed networks", Journal of Parallel and Distributed Computing, Vol. 9, No. 4, 2000, pp. 374-386.

[8] Yang, C. C. Chen, C. K. Chang, Y. H. Chung, K. H. Lee, J. K., "Software architecture design for streaming Java RMI", Science of computer programming, Vol.70 No.2-3, 2008, pp.168-184.

저자 소개

허 진 경(정회원)

- 2004년 2월 조선대학교 전산통계학과(박사)
- 2006년 3월 ~ 2008년 8월 호원대학교 사이버수사경찰학부 연구교수
- 2008년 9월 ~ 2010년 8월 호원대학교 사이버수사경찰학부 전임강사
- 2010년 9월 ~ 현재 : 호원대학교 사이버수사경찰학부 조교수

<주관심분야 : 컴퓨터통신, 인터넷통신, 분산처리>

※ 본 논문은 2010년 호원대학교 교내학술연구조성비에 의해 연구되었음.