분산 시각미디어 검색 프레임워크를 위한 모니터링 시스템

심준용° 김세창 원재훈 김정선
한양대학교
{jyshim°, sckim, jhwon, jskim}@cse.hanyang.ac.kr

The Monitoring System for Semantic Web based Visual Media Retrieval Framework

Junyong Shim°, Sehchang Kim, Jaehoon Won, Jungsun Kim
Dept. of Computer Engineering, Hanyang University

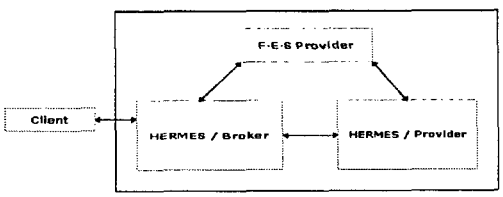
요 약

기존의 Ontology를 이용한 이미지 검색 시스템이나, 간단한 구조를 가진 메타데이터 기반의 분산 이미지 검색 시스템들의 단점들을 극복하기 위해 다양한 이미지 제공자들의 자율성을 보장하면서, 시맨틱 기반의 이미지 검색을 지원하는 분산 시각미디어 검색 프레임워크가 제안되었다. 하지만 제안된 프레임워크에서는 Visual Media Data를 제공하는 Provider와 클라이언트의 Query를 처리해서 Provider에게 전달하는 Broker 사이의 연결 상태에 대한 신뢰성이 보장되지 않았고, 다수의 클라이언트 접속에 의해 발생하는 Broker 내부 컴포넌트들의 Overhead 문제를 효과적으로 해결할 수 없었다. 본 논문에서는 기존의 프레임워크에 Monitoring 시스템을 도입하여 Broker내부 컴포넌트들의 수행시간을 측정하여 저장함으로써, 다수의 클라이언트 요구에 의해서 Overhead가 발생하는 컴포넌트들을 Monitoring 할 수 있고, Provider의 연결 상태를 정기적으로 확인하여 Broker내부에 등록되어 있는 Provider의 도메인 리스트를 서버 상태가 확인된 리스트로 업데이트 시켜줌으로써 연결 상태에 대한 신뢰성을 제공할 수 있도록 하기위한 Monitoring 시스템을 제안한다.

1. 서론

기존의 Ontology를 이용한 이미지 검색 시스템이나, 간단한 구조를 가진 메타데이터 기반의 분산 이미지 검색 시스템들의 단점들을 극복하기 위해 다양한 이미지 제공자들의 자율성을 보장하면서, 시맨틱 기반의 이미지 검색을 지원하는 프레임워크인 HERMES(The Retrieval Framework for Visual Media Service)가 연구되었다.[1][2]

HERMES 아키텍쳐는 HERMES/B, HERMES/P, F·E·S Provider와 같은 세 개의 주요 컴포넌트들로 구성 되어있다. 주요 특징을 살펴보면 각 컴포넌트들 간의 플랫폼 독립적인 웹서비스를 이용하여 이기종간의 서비스를 가능하게 하였고, 메타데이터와 Ontology를 이용한 지능적인 Visual Media data 검색을 지원했다. 기존 아키텍쳐는 아래 [그림 1]과 같다.



[그림 1] HERMES Architecture

하지만 제안된 프레임워크는 클라이언트가 원하는 Visual Media Data를 제공하는 Provider와 클라이언트의 Query를 처리해서 Provider에게 전달하는 Broker 사이에 연결 상태에 대한 신뢰성이 보장되지 않았고, 다수의 클라이언트 접속에 의해 발생하는 Broker 내부 컴포넌트들의 Overhead에 대한 문제를 효과적으로 해결할 수 없었다.

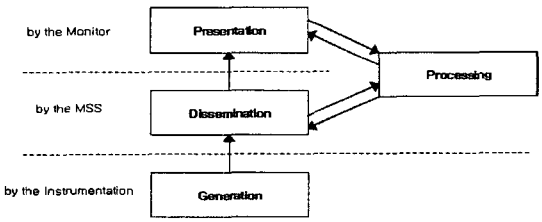
본 논문에서는 기존의 제안된 프레임워크가 가지고 있는 단점들을 보완하기 위해서 Broker 내부의 컴포넌트들의 수행시간[3]을 측정함으로써 서버의 증설 전략이나 컴포넌트들의 워크플로우를 최적화 시킬 수 있는 전략을 계획하여 다수의 클라이언트들의 요구에 의한 Broker 서버의 Overhead를 줄일 수 있고, Provider 서버의 연결 상태를 Ping을 통하여 정기적으로 측정하여 문제가 생긴 서버에 대한 Provider 리스트를 Broker의 ServiceOntology에 업데이트 하여 Broker와 Provider의 연결 상태의 신뢰성을 증가시킬 수 있는 Monitoring 시스템을 제안한다.

2. 기반기술

2.1 Monitoring의 activities

Monitoring 시스템들은 [그림 2]의 주요 동작을 갖게 된다.[4][5]

2.1.1 Generation
오브젝트들에 대한 정보 수집
2.1.2 Processing
수집되어진 정보들이 요구된 format으로 생산되어지는 과정
2.1.3 Dissemination
전 단계의 정보들이 그것들을 요구하는 user들에게 전달되어지는 과정
2.1.4 Presentation
모여진 정보들이 user가 원하는 format(텍스트, 그래프 등)으로 표현



[그림 2] Functional model for Monitoring

2.2 컴포넌트의 객관적인 성능평가를 위한 요소

컴포넌트의 성능은, 컴포넌트를 사용하는데 소비된 비용을 수치화된 데이터로 표현하는 것이다. 소비된 비용이란, 컴포넌트의 기능을 사용할 때 소비된 시간 값인 시간적인 비용과 컴포넌트의 사용이 시스템에 영향을 주는 정도인 시스템 리소스의 소비를 포함한다. 컴포넌트의 객관적인 성능평가를 위한 성능측정 요소는 다음과 같다.[3]

2.2.1 메소드별 응답시간(메소드 Response Time)
메소드별 응답시간은 컴포넌트의 단위기능의 응답시간이다. 이 요소는

사용자가 요구하는 컴포넌트의 기능별 성능을 나타내는 기준이 된다.

2.2.2 컴포넌트 응답시간(Component Response Time)

컴포넌트의 응답시간은 컴포넌트 인스턴스 생성 시간을 포함한 컴포넌트 내의 모든 메소드의 응답시간의 합이다. 이것은 컴포넌트의 전체의 성능을 평가할 수 있는 요소로서의 의미를 가진다.

2.2.3 CPU 사용률(CPU Usage Rate)

CPU사용률은 컴포넌트가 호출됨으로써 생기는 CPU사용률의 변화량이다. CPU사용률은 컴포넌트를 사용하는 것이 시스템에 어떠한 영향을 주는지를 판단하기 위한 성능요소이다.

2.2.4 메모리 사용률(Memory Usage Rate)

메모리 사용률은 컴포넌트의 메소드를 사용할 때의 메모리 사용률의 변화량을 의미한다. 메소드 사용률은 컴포넌트의 사용이 시스템의 메모리에 어떠한 영향을 주는지 판단하기 위한 성능요소이다.

2.3 분산 환경에 대한 Monitoring System의 역할

2.3.1 테스트와 유효성검사

분산 환경에 있는 컴포넌트들의 런 타임 시 행위에 대하여 테스트하고 유효한지를 검사하는 역할로, 이벤트 검사, 결과가 Prototype과 일치하는지 테스트, 에러부분 탐지 등이 있다.

2.3.2 동작과 유지보수

컴포넌트들의 동작과 그것의 결과에 대한 오류탐지 및 유지보수를 하는 역할로, 동작에 대한 오류탐지, Fault 대한 유지보수, 재구성 관리, 계정 관리, 성능 관리 그리고 보안에 대한 관리 등이 있다.

2.4 Monitoring의 방법

Monitoring의 방법에는 Event가 발생되어졌을 때 즉각적으로 보여줄 수 있는 Event-driven방법과 Monitoring 되는 데이터를 시간에 따라 주기적으로 수집하여 정보를 얻을 수 있는 Time-driven방법이 있다.

2.4.1 Event-driven 방법

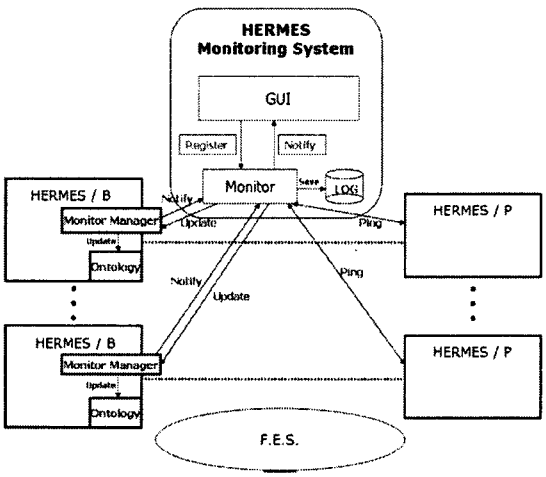
- 관심을 두고 있는 사건들의 발생의 정보를 얻는데 기반을 둔 방법.
- 수집되어 지는 시스템의 변화에 대한 정보에 의해 어플리케이션 활동에 대한 동적인 View를 제공
- Time-driven 방법과는 다르게 시간적으로 주기적인 정보를 얻는 것이 아니기 때문에 Overhead가 적음

2.4.2 Time-driven 방법

- 주기적인 상태 정보를 얻는데 기반을 둔 방법.
- Monitoring되는 오브젝트의 행위의 누적된 데이터를 제공하는 방법
- Sampling되는 비율과 제공되는 정보의 양이 직접적인 관계가 있는 방법[4][6][7]

3. Architecture

본 논문에서 제안한 Monitoring System Architecture는 [그림 3]과 같다.



[그림 3] Monitoring System Architecture

3.1 Monitoring System 구성요소

Broker의 수행시간을 측정하여 저장하고, GUI에 실시간으로 Broker의 컴포넌트 상황을 시각적으로 보여준다. 또한 Provider들에게 Ping을 이용하여 각 Provider들의 서버 상태에 대한 정보를 수집하여 Broker의 ServiceOntology의 도메인 리스트를 업데이트 한다.

3.1.1 GUI 컴포넌트

- Broker의 워크플로우 상태를 실시간으로 보여주고, 저장되어 있는 각 컴포넌트들의 수행시간을 시간대별로 그래프를 통하여 보여준다.
- 체크된 Provider의 서버상태를 화면에 보여준다.

3.1.2 Monitor 컴포넌트

- Broker의 각 컴포넌트의 수행시간을 측정하여 저장하고, 화면에 나타내기 위해 GUI컴포넌트에 이벤트를 발생시킨다.
- 각 Provider의 서버들에게 Ping을 사용하여 서버들의 상태정보를 체크하고, 그 정보를 가지고 Broker의 ServiceOntology의 도메인 리스트를 업데이트한다.

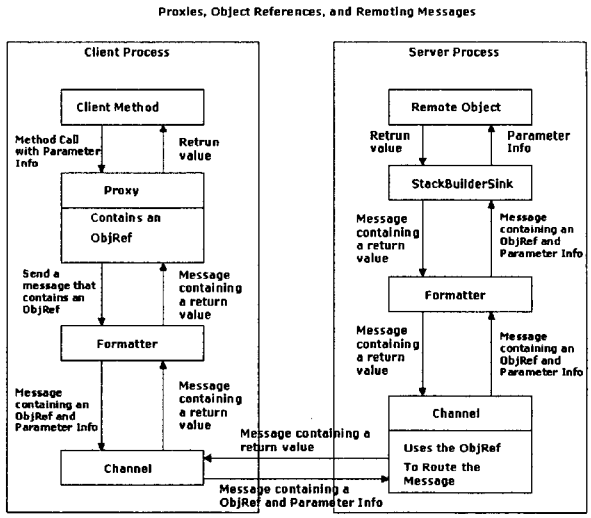
3.2 Monitoring System 구현기술

3.2.1 상태측정을 위한 Ping명령

- Monitoring System의 Monitor 컴포넌트는 Broker의 ServiceOntology로부터 각 Provider의 도메인 리스트를 얻어온 후, 그 도메인을 가지고 각 Provider들에게 Ping명령을 사용한다. 그것에 대한 응답을 하는 서버들과 응답하지 않은 서버들을 구분하여 GUI컴포넌트에게 알리고, 응답하지 않은 서버들을 제외시킨 도메인 리스트를 Broker의 ServiceOntology에 다시 업데이트한다.

3.2.2 분산 Monitoring을 위한 .Net Remoting

- .Net Remoting의 아키텍쳐는 아래 [그림 4]과 같다.[8][9]



[그림 4] .Net Remoting Architecture

- 개념 및 구성요소

클라이언트 프로그램이 원격 오브젝트의 메소드를 호출할 수 있게 해주는 기술로 Remote Object, Channel, Formatter의 세 가지 주요 요소들을 가지고 있다.

- 제안된 Monitoring System은 Remote Object로 시간을 측정해줄 수 있는 NotifyManager라는 컴포넌트를 만들고, Channel은 HTTP Channel과 Formatter는 Soap Formatter를 사용하여 3.1에서 설명한 기능을 구현하였다.

4. Monitoring System의 특징

본 논문에서 제안하는 Monitoring System은 중요한 두 가지 기능을 지원하고 있다.

4.1 Ping을 사용한 Provider의 상태정보 제공

HERMES Broker는 등록된 Provider의 도메인 리스트를 저장하고 있는 ServiceOntology 컴포넌트를 가지고 있다. 하지만 등록된 Provider의 도메인 리스트만을 알고 있을 뿐 Provider의 연결 상태나 서버의 상태 정보는 알 수 없다. 다시 말해서, 클라이언트는 Broker를 통해서 Provider로 부터의 서비스를 받기 때문에 기존의 제안된 Broker만을 가지고 Provider와의 링크 상태나 서버의 상태에 대한 정보를 알 수 없다.

따라서 Provider가 불안정한 링크 상태를 가지고 있거나 서버에 문제가 생겼을 경우에도 Broker 서버는 이미 등록되어진 Provider의 도메인 리스트만을 가지고 클라이언트의 Query를 해당 Provider에게 보낸다. 즉, Query가 수행되어질 시점에서는 Provider의 상태가 고려되어지지 않는 것이다.

이것은 문제가 생긴 Provider에게도 클라이언트의 요구가 전달되어지기 때문에 클라이언트에게 긴 응답시간을 가져다주게 된다. 따라서 이러한 문제를 해결하기 위해서 본 논문에서 제안한 Monitoring System은 Broker 서버로부터 현재 등록된 도메인 리스트를 가져와서 리스트에 있는 각각의 Provider에게 Ping명령을 사용하여 Provider들의 상태정보를 파악한다.[그림 5]

이 명령에 응답을 하지 못하는 Provider 서버는 클라이언트의 요구에 응답을 할 수 없는 상태이기 때문에 이를 Broker 서버에게 알림으로써 클라이언트의 Query를 각각의 Provider에게 보내는 경우 문제가 생긴 Provider를 제외시키도록 한다.[그림 6]

즉, Broker에 등록되어있는 Provider인 경우라도 문제가 생긴 Provider에게는 클라이언트의 Query에 대한 요청을 하지 않으므로 응답을 할 수 없는 Provider로부터 발생한 지연시간을 제거하게 된다.

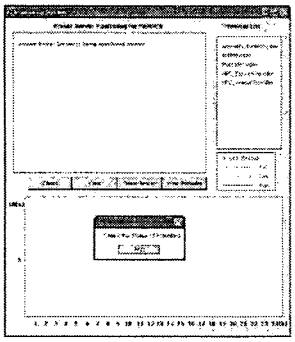
이는 Broker 서버의 성능향상을 가져오며, 클라이언트의 요청에 대한 응답시간을 줄여주게 된다.

4.2 Broker내부의 컴포넌트의 Monitoring을 통한 정보제공

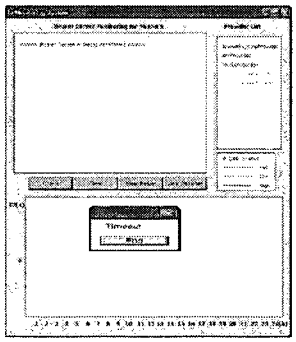
3.2에서 기술한 .Net Remoting 기술을 사용해서 Broker내부의 컴포넌트들 간의 수행시간을 측정하여 저장할 수 있고, 저장된 데이터와 각 컴포넌트들의 호출 순서를 GUI화면을 통해서 확인할 수 있게 된다.[그림 7][그림 8] 제안한 Monitoring 시스템에서 측정하고 있는 컴포넌트의 타임정보는 클라이언트의 Query가 Broker의 전체 컴포넌트들을 수행한 시간과, 각각의 컴포넌트의 수행시간을 측정한 시간이다.

이렇게 측정한 데이터를 측정 당시의 시간과 함께 저장하여 GUI화면에 각 시간대별로 측정 시간대별 평균을 그래프로 보여준다.[그림 7][그림 8] 관리자는 저장된 데이터로부터 Overhead에 대한 정보를 확인하여 Broker서버의 증설 여부를 판단할 수 있고, 다수의 Broker 서버를 관리할 경우 클라이언트의 요청에 대한 부하균등 정책을 계획할 수 있다.

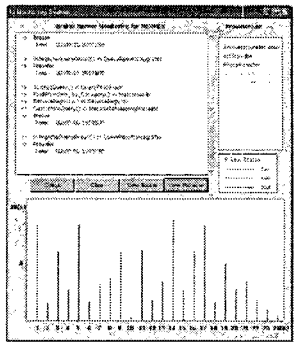
이는 클라이언트에게 Broker서버의 안정성에 대한 신뢰를 더 높여준다. 또한 각각의 컴포넌트들의 호출 관계와 수행시간을 GUI화면에 실시간으로 보여줌으로써 각 메소드의 Overhead정보와 문제가 발생했을 시 발생된 컴포넌트를 알 수 있게 한다.



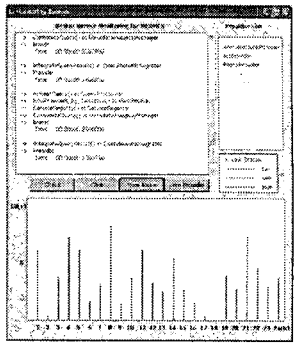
[그림 5] Monitoring System 초기화면



[그림 6] Domain List에 있는 Provider 서버상태 측정



[그림 7] Provider 서버 측정 결과 화면



[그림 8] Broker 서버 측정 결과 화면

5. 결론 및 향후 과제

본 논문에서는 기존의 HERMES 아키텍쳐에서 고려하지 않았던 Provider의 상태정보를 Monitoring 하고, Broker의 각 컴포넌트들의 수행시간을 측정함으로써 클라이언트 요구의 응답에 대한 신뢰성을 높이며, 관리자로 하여금 Broker 서버의 안정성을 위한 정책을 계획할 수 있는 정보를 제공하였다.

하지만 Provider의 상태정보만을 Monitoring해서 클라이언트가 요구하는 컨텐츠에 대한 신뢰성은 높였다고 단정 지을 수 없다. 이를 위해 Provider가 가지고 있는 각 컨텐츠들의 Quality를 측정할 수 있는 기준이나 요소들이 연구되어야 한다. 또한 Broker 서버를 증설할 경우, 하나의 서버 컴포넌트 내부만을 측정한 데이터 외에 Broker 서버간의 필요한 측정요소가 있는지에 대한 연구가 필요하다.

6. 참고 문헌

[1]양영미, 정병훈, 손영수, 김정선, "Ontology를 이용한 Web Services 기반 분산 이미지 검색 프레임워크 Architecture", 한국컴퓨터종합학술대회 2005 논문집 Vol.32, No.1(B)

[2]나연묵, 이복주, 김정선, "Visual Media Retrieval Framework Using Web Service", Lecture Notes in Computer Science 3597권 pages 104-113, Jul, 2005

[3]황길승, "미들웨어 중립적인 컴포넌트 성능측정도구 설계" 한국항공대학교 컴퓨터공학과, 2003년도 한국정보과학회 봄 학술발표논문집 Vol.30, No. 1

[4]Masoud Mansouri-Samani and Morris Sloman, "Monitoring Distributed Systems", IEEE Network, November, 1993

[5]Nikolay Diakov, Monitoring Distributed Object And Component Communication, pages 45-46 The Netherlands, 2004, ISBN 90-75176-38-4

[6]High-performance Monitoring Architecture for Large-scale Distributed Systems Using Event Filtering Ph.D. Proposal, http://www.mnlab.cs.depaul.edu/~ehab/proposal

[7]Scarlet Schwiderski, "Monitoring the Behaviour of Distributed Systems", Selwyn College, University of Cambridge, April, 1996

[8]David Conger, Remoting with C# and .NET, Wiley Publishing, Inc., 2003, ISBN 0-471-27352-X

[9]Microsoft's .net Remoting: A Technical Overview, http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnotnet/html/hawkremoting.asp