차세대 웹 환경에서 Complex Event Processing 엔진을 이용한 대용량데이터 처리

(High-Volume Data Processing using Complex Event Processing Engine in the Web of Next Generation)

강 만 모 † 구 자 록 † 이 동 형 ††
(Manmo Kang) (Rarok Koo) (DongHyung Lee)

요 약 웹이 성장함에 따라 데이터처리 기술도 발전하고 있다. 차세대 웹 환경에서는 다양한 유무선 사용자, USN, RFID를 위한 고속, 대용량데이터 처리기술 또한 발전하고 있다. 본 논문에서는 CEP(Complex Event Processing) 엔진을 이용하여 대용량데이터를 처리하는 기술을 제안한다. CEP는 복잡한 이벤트를 처리하는 기술로 CEP 엔진은 다음과 같은 특징이 있다. 첫째 대용량의 이벤트(데이터)를 받는 작업, 둘째 이를 분석하는 작업, 최종적으로 새로운 액션으로 연결시키는 작업으로 나눌 수 있다. 즉 대용량데이터를 수집하고 이벤트들을 분석, 필터링한다. 또한 이벤트 엔진에 미리 등록해 놓은 이벤트와 새로운 이벤트를 패턴매칭하여 데이터를 추출한다. 추출된 결과를 다른 작업의 입력 이벤트로 사용하거나 요청된 이벤트에 대해 실시간으로 응답할 수 있고 유효한 데이터만 데이터베이스에 트리거할 수도 있다.

키워드 : 대용량데이터, 복잡한 이벤트처리, 실시간

Abstract According to growth of web, data processing technology is developing. In the Web of next generation, high-speed or high-volume data processing technologies for various wire-wireless users, USN and RFID are developing too. In this paper, we propose a high-volume data processing technology using Complex Event Processing(CEP) engine. CEP is the technology to process complex events. CEP Engine is the following characteristics. First it collects a high-volume event(data). Secondly it analyses events. Finally it lets event connect to new actions. In other words, CEP engine collects, analyses, filters high-volume events. Also it extracts events using pattern-matching for registered events and new events. As the results extracted. We use it by an input event of other work, real-time response for demanded event and can trigger to database for only valid data.

Key words : high-volume data, complex event processing, real-time

1. 서 론

차세대 웹의 주요 키워드는 실시간, 개인화 그리고 서비스이다. 시맨틱 웹, 분산 웹서비스와 Open API, 모바일 웹, 각종 스마트 디바이스 등의 차세대 웹 환경에서는 실시간 대용량데이터를 처리하는 기술이 요구된다. 네트워크에 분산된 다양한 서비스의 융복합을 실현하기 위한 미래형 웹 기술에는 동적 서비스를 제공하는 유비쿼터스 웹 기술, 유무선 웹 콘텐츠를 통합하고 새로운 비즈니스를 창출하기 위한 모바일 웹 기술, 사용자에게 보다 풍부한 웹 사용 환경을 제공하는 리치웹 같은 웹 2.0 기술 그리고 차세대 컴퓨팅 플랫폼을 위한 웹 기반 플랫폼 기술 등이 있다[1]. 이러한 차세대 웹 환경에서 다양한 서비스를 제공하기 위한 실시간 대용량데이터 처리는 필수 요건이라 할 수 있다. IBM의 WebSpere, Oracle의 WebLogic, Progress Software의 Apama, Sybase의 Coral8, TIBCO의 TIBCO BusinessEvents, StreamBase Systems의 Event Processing Platform 등 국내외 IT 업체는 대용량데이터를 처리하기 위해

† 정 회 원 : 울산대학교 컴퓨터정보통신공학부 교수
manmoakng@ulsan.ac.kr
koorok@ulsan.ac.kr
†† 정 회 원 : 한국폴리텍VII대학 정보통신시스템과 교수
sunhook@dreamwiz.com
논문접수 : 2010년 8월 20일
심사완료 : 2010년 9월 24일

정보과학회논문지: 데이타베이스 제37권 제6호(2010.12)

다양한 제품들을 출시하고 있다. 이러한 제품들은 실시간 대용량의 데이터 처리하기 위해 ESP(Event Stream Processing) 기술 및 CEP(Complex Event Processing) 기술을 이용하고 있다. 본 논문의 구성은 다음과 같다. 2장에서는 복잡한 이벤트를 처리하기 위한 관련연구를 소개하고 3장에서는 대용량데이터 처리 모델을 설계하고 4장에서는 설계모델을 기반으로 프로토타입을 설계 및 구현하고 5장에서는 결론 및 향후과제를 기술하고 6장에서는 응용분야에 대하여 알아본다.

2. 관련연구

2.1 Simple Event Processing, ESP, CEP

이벤트를 처리하는 방법에는 크게 3가지로 나눌 수 있다. 중소규모의 이벤트를 처리하는데 사용하는 간단한 이벤트 처리(Simple Event Processing)와 복잡하고 대용량의 이벤트를 처리하는데 사용하는 이벤트 스트림 처리(Event Stream Processing, ESP)과 복합 이벤트 처리(Complex Event Processing, CEP)가 있다[2].

ESP와 CEP는 대용량의 데이터를 처리하는 관점에서 비슷한 기술로 생각할 수 있다[3]. ESP는 1990년대 중반 DBMS 업체들을 중심으로 실시간 데이터를 분석하여 연속적으로 발생하는 스트림 이벤트를 처리하기 위한 기술로서, 시간의 경과에 따라 변화하는 이벤트들의 집합인 스트림을 빠른 속도로 분석하는 기술에 중점을 두고 있다. ESP 기술을 상업적으로 처음으로 적용한 대상은 알고리즘트레이딩을 위한 마켓 데이터분석이며 주로 주식, 환율, 유가시장에 많이 사용한다[3]. CEP기술은 1980년 후반부터 분산 환경에서 시스템들이 발생시키는 이벤트들을 분석하기 위해 등장했으며 BAM(Business Activity Monitoring)에 적용되었다. CEP는 다양한 장소, 장비 및 서비스에서 발생하는 이벤트들의 집합인 클라우드(Cloud)에서 수집한 이벤트에 대한 특정한 패턴 정보를 분석, 추출 및 필터링하여 실시간으로 응답할 수 있는 기술에 중점을 두고 있다[3,4]. CEP는 실시간 대용량의 데이터처리를 요구하는 RFID, 각종 센서 및 다양한 서비스를 요구하는 분야에서 사용할 수 있다.

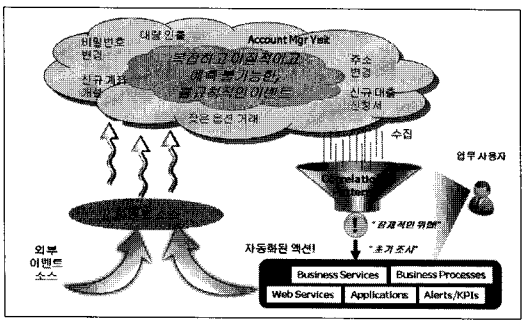


그림 1 CEP의 개념적인 구조

2.2 SOA와 EDA

SOA(Service Oriented Architecture)에 대한 정의는 서로 다르지만 SOA는 '공유할 수 있고, 재사용할 수 있는 서비스 컴포넌트 기반 정보시스템을 구축하는 아키텍처 모델'이라고 할 수 있다. SOA의 관점에서 서비스는 독립적(self-contained)이고, 인터페이스를 통해 자신이 가진 비즈니스 프로세스를 처리할 수 있는 컴포넌트들로 정의되며, 비즈니스 프로세스와 서비스는 시간이 흐르면서 변하는 특성을 갖는다[5].

SOA와 더불어 최근에는 EDA에 대한 관심 또한 커지고 있고 SOA와 EDA를 합하여 SOA 2.0이라고도 명명하고 있다[6]. 정보 조사 기관에 의하면, 현재는 대기업과 중소기업이 SOA와 EDA 기반으로 응용을 구축하고 있고, 패키지 응용에서도 적용하고 있다[6]. SOA와 EDA는 상호 보완적인 서비스 구축 방법론으로 SOA의 기본 개념이 정의된 서비스 인터페이스를 이용하여 요청(request) 및 응답(response)을 통해 서비스를 연동하는 것이라면, EDA는 그림 2와 같이 이벤트에 대한 감지(sense) 및 대응(respond) 모델이다. 표 1과 같이 SOA는 클라이언트에 의해 서비스가 제어되며 순차적으로 실행되는 반면, EDA는 이벤트 수신자가 대응 여부를 결정하며, 이벤트가 동시에 여러 곳으로 전달이 가능하고 또한 비동기 방식으로 전달 가능하므로 이벤트 발생에 의한 대응이 동적으로 구성된다.

예를 들어, 쇼핑몰에서 클라이언트가 물건을 산다고

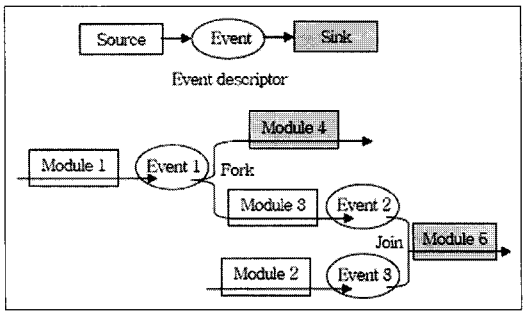


그림 2 EDA

표 1 SOA와 EDA 비교

구분	SOA	EDA
상호규약정보	서비스 인터페이스 정보	이벤트 규격 정보
연결방식	1:1	N:N
흐름제어주체	클라이언트	이벤트 수신자
흐름제어방식	순차경로	진행 중에도 반응
새로운 입력에 대한 대응	진행 중엔 차단	동적/병렬/비동기

하면, SOA의 경우 클라이언트는 필요한 서비스를 찾고, 서비스 제공자에게 서비스를 요청하고, 요청에 대한 응답을 받는 단계를 순차적으로 처리한다. 즉 서비스 요청자와 제공자가 사전에 표준화된 호출 규약의 합의에 의해 순차적인 경로로 진행하므로 하나의 서비스가 다른 서비스의 상태에 의존하게 된다. 반면, EDA의 경우 클라이언트는 다수에게 이벤트를 발송하기만 하면 되고, 이벤트의 처리에 대한 내용은 클라이언트가 전혀 관여하지 않는다. 즉 클라이언트와 수신자가 서로에 대한 규약 없이 독립적으로 수행된다. 따라서 EDA가 보다 향상된 유연성을 제공할 수 있다.

양 서비스 구축 방법론의 등장은 구축하고자 하는 서비스 자체의 속성에 따른 차이를 반영한 것 뿐만 아니라, SOA가 서비스의 재활용 및 공유가 주목적이라면, EDA는 실시간 서비스 제공이 주목적이라고 할 수 있다[5,6].

2.3 ESB

ESB(Enterprise Service Bus)는 SOA를 따르는 가장 대표적인 미들웨어 플랫폼으로, 분산된 서비스를 조합하여 애플리케이션을 구성하도록 지원하는 SOA의 개념을 실현하는 중요한 기술이다[7]. ESB는 웹 서비스, 컨텐츠 기반의 라우팅, 메시지 변환 기술을 기반으로 B2B 내의 서비스, 애플리케이션, 자원들을 연결 및 통합하는 기능을 수행한다. ESB는 메시지 기반의 느슨하게 결합한 표준 인터페이스 기반 통신을 지원하고 있으며, 표준적인 메시지 SOAP을 지원하며 서비스의 조합과 실행을 위한 플로우 제어에 의한 통합을 지원한다. 이를 통해 ESB는 애플리케이션의 유연성과 민첩성을 보장해 주며, 복합 서비스 구성과 유연한 서비스 관리기술을 제공한다. 현재 ESB는 필요한 기능을 선별하고 특화하여 정식적인 솔루션으로 출시되어 있고 또한 ESB를 구현한 많은 오픈소스 프로젝트들도 진행되고 있다.

그림 3은 C/C++로 구현된 레거시 애플리케이션, ERP 시스템, NET와 J2EE 플랫폼 그리고 웹 서비스들이 신뢰성 있는 비동기식 보안 메시지 전송인 자바 메시징 서버(Java Messaging Server)를 통해 통합되는 ESB 구조이다.

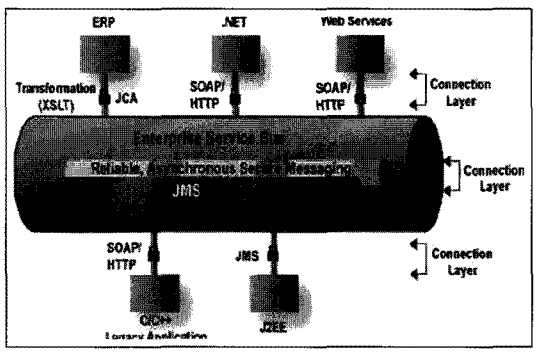


그림 3 ESB

2.4 복합이벤트처리 엔진과 이벤트처리언어

2.4.1 관계형 데이터베이스

차세대 웹 환경은 실시간 대용량데이터 처리를 요구한다, 그러나 현존하는 관계형 데이터베이스로 처리하는데 몇 가지 어려움이 있다. 첫째 관계형 데이터베이스와 SQL은 대부분 정적인 데이터를 처리하도록 설계되어 있어 복잡한 질의에 대해 동적으로 반응하기 어렵다. 둘째 인-메모리(In-memory) 데이터베이스를 제외한 대부분의 데이터베이스는 디스크를 액세스하는데 최적화되어 있다. 셋째 비록 데이터베이스 트리거가 메시지들을 한 단위로 묶어서 데이터베이스의 업데이트 이벤트 응답에 사용될 수 있지만 데이터베이스 트리거는 속도가 느리고 복잡한 상태체킹 수행 및 실시간으로 반응하는 로직을 구현하기 쉽지 않다. 마지막으로 인-메모리 데이터베이스가 속도적인 면에서 CEP 애플리케이션보다 효율적일 수도 있지만 복잡하고 연속적인 이벤트 분석을 통한 실시간 결과를 요구하는 것에는 최적화되어 있지 않다.

2.4.2 CEP 엔진

제안하는 CEP 엔진은 다음과 같은 특징을 가지고 있다. 첫째 실시간 대량 데이터 수집, 생성가능하다. 둘째 쉬운 데이터 분석, 필터링하여 로직을 구현할 수 있다. 셋째 대용량데이터 처리에 효율적인 이벤트 기반 구조를 가진다. 넷째 고속, 대용량의 데이터처리 환경에서 원활한 서비스 및 실시간 데이터 추출 및 처리가 가능하다. 마지막으로 서비스 유형에 따른 데이터 분산, 취합 프로세스를 제공한다. 이와 같은 이유로 데이터베이스 관련업체들이 CEP 엔진에 주목하고 있다.

2.4.3 EPL

데이터베이스가 데이터를 처리하기 위해 DBMS를 사용한다면 CEP는 CEP 엔진을 사용한다. CEP 엔진은 복잡한 이벤트(데이터)를 수집하여 분석 및 필터링하여 미리 등록한 이벤트와 새로운 이벤트의 패턴 매칭을 통해 의미있는 이벤트만 추출하여 실시간으로 출력할 수 있다. 데이터베이스가 데이터를 추출하는데 SQL를 사용한다면 CEP는 EPL(Event Processing Language)를 사용한다. EPL은 문법적으로 SQL과 비슷한 구조를 가지며 아직 완전한 표준은 없고, CEP 기술을 사용하는 회사들마다 약간의 차이가 있다. EPL의 Event Stream Views는 SQL의 Tables와 비슷하고, Events는 Rows와 비교할 수 있다. 본 논문에서 이용한 Esper 엔진(CEP 엔진)이 제공하는 명령문은 다음과 같다[8].

- 타임 윈도우(time window)

select * from Withdrawal.win:time(4 sec)
Withdrawal 이벤트를 4초 동안 관찰한다.

- 데이터 윈도우(data window)

select * from Withdrawal.win:length(5)
길이가 5인 length window에 Withdrawal 이벤트를 넣고, 결과를 UpdateListener에게 알린다.

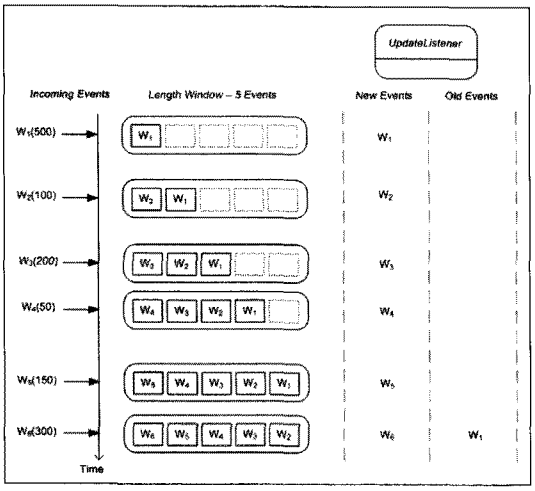


그림 4 데이터 윈도우 예

- 필터링(Filtering)

select * from Withdrawal(amount>=200).win:length(5)
길이가 5인 length window에 이벤트 Withdrawal의 amount가 200보다 크거나 같은 이벤트만 넣고 이를 UpdateListener에게 알린다.

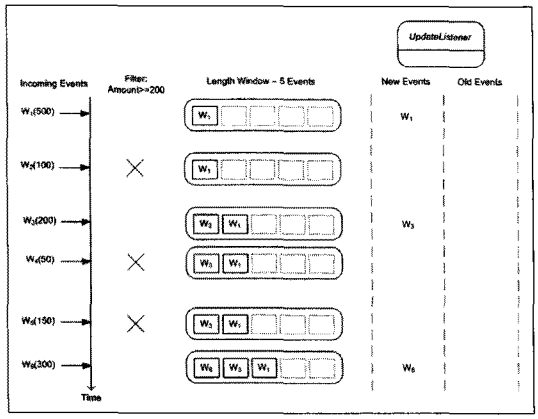


그림 5 데이터 윈도우에서 필터링 예

- 패턴 매칭(pattern matching)

select tick.symbol as symbol, tick.price as price from pattern[every tick=StockTick where timer:within(10 sec)]
10초 이내의 모든 StockTick 이벤트패턴과 매칭되는 종목(symbol)과 가격을 보여라.

- 이벤트 집계

select sum(amount) from Withdrawal.win:time(5 sec)
5초 동안 이벤트 Withdrawal의 amount 합계

- 이벤트 결합

select customerId, customerName from CustomerCallEvent.win:time(30 sec) as cce, sql:MyCustomerDB ["select cust_id as customerId, cust_name as customerName from Customer where cust_id = ${cce.customerId}"] as cq
30초 동안 스트림 이벤트 CustomerCallEvent와 데이터베이스(MyCustomerDB)의 테이블(Customer) 의한 결합(customerID=cust_id)

- 입력 및 출력

insert into SMAStream select sensorID, avg(temperature) as temperature from SensorEvent.win:time(1 days) group by sensorID output all every 60 minutes
1일 동안 SensorEvent에서 sensorID와 평균기온(avgtemperature)을 SMAStream 이벤트에 60분 단위로 저장한다.

3. 대용량 데이터 처리

3.1 통합 프레임워크

본 논문에서 제안하는 대용량데이터를 처리하기 위한 전체적인 구조는 그림 6과 같다. SOA 기반 서비스는 다수의 사용자, RFID 리더기, 다양한 센서들 그리고 모바일 폰(스마트 폰)등 입력받은 대용량데이터를 수집한다. 이러한 데이터들은 다양한 프로토콜을 지원하는 ESB을 통해 EDA에 전달한다. EDA는 연속적인 요청 및 질의를 받게 되고, 연속적인 질의는 이벤트 생성기를 통해 이벤트가 생성되고 이벤트 엔진에 이들을 전달한다.

CEP 엔진은 복잡한 이벤트를 분석, 필터링, 결합(Join), 집계(Aggregation), 패턴매칭을 사용하여 매칭되는 데이터를 요청한 사용자에게 실시간으로 응답하거나 연관된 응용프로그램(모니터링 시스템, 예측시스템 또는 웹 서버)의 데이터로 실시간 활용될 수 있고, 미리 등록한 패턴과 일치하는 데이터를 데이터베이스에 저장할 수도

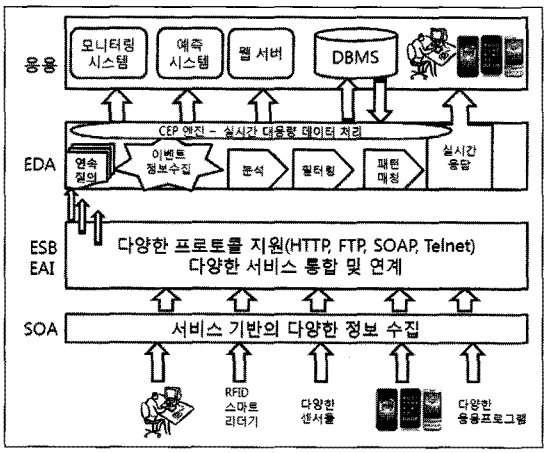


그림 6 통합 프레임워크

있다. 데이터베이스는 매칭되지 않는 이벤트와는 관계하지 않으며 유효한 데이터만을 저장함으로서 데이터베이스의 부하를 줄일 수 있다.

3.2 이벤트 처리 과정

이벤트 처리 과정은 전사적인 관점에서 크게 3단계로 나눌 수 있다. 첫째 이벤트들을 받는 단계, 둘째 이벤트들을 받고 분석하고 걸러내는 단계, 최종적으로 다시 새로운 액션으로 연결시키는 단계로 나눌 수 있다.

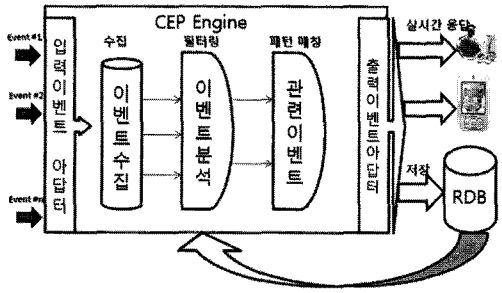


그림 7 이벤트 처리 과정

그림 7의 이벤트 처리과정을 살펴보면 다음과 같다.

- 입력 이벤트 아답터

사용자, RFID, Sensor, 모바일 장치 등의 요청이 입력되면 이벤트 생성기를 통해 이벤트를 생성한다.

- 이벤트 수집

생성된 복잡한 이벤트를 수집한다.

추출된 결과를 저장하고 있는 데이터베이스의 데이터도 다시 수집하여 입력 이벤트로 사용할 수도 있다.

- 이벤트 분석

결합(Join), 집계(Aggregation), 필터링(Filtering)을 통해 이벤트를 분석한다.

- 관련 이벤트

미리 이벤트 엔진에 등록되어 있는 이벤트와 새로운 이벤트를 패턴매칭하여 이벤트를 추출하여 결과를 출력 이벤트로 전달한다.

- 출력 이벤트 아답터

서비스를 요청한 사용자 및 응용프로그램에 실시간으로 응답하거나 그 결과를 데이터베이스에 직접 저장하거나 트리거할 수도 있다.

4. 프로토타입 설계 및 구현

프로토타입 설계를 위해 CEP 엔진으로 Esper를 사용하였으며 언어는 Java를 사용하였다.

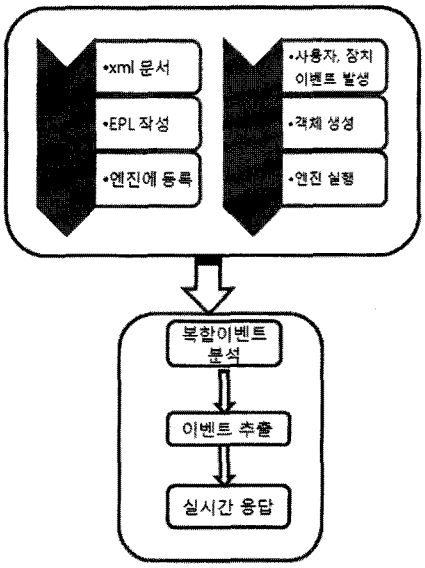


그림 8 애플리케이션 전체 구조

- 이벤트 객체 생성

사용자, 센서 RFID, Mobile 등 이벤트 객체를 생성한다.

예) UserEvent, SensorEvent, RFIDEvent, MobileEvent

- 리스너 객체 생성

사용자, 센서 등을 감시하는 리스너 객체를 생성한다.

예) UserListener, SensorListener, RFIDListener, MobileListener

- CEP 엔진 생성 및 이벤트 리스너에 등록

CEP 엔진을 생성하고 이벤트 리스너에 등록한다.

예) UserStatement, SensorStatement,
RFIDStatement, MobileStatement
사용자 이벤트, 센서 이벤트 등을 모니터링한다.
이벤트 패턴을 등록하여 해당 패턴을 추출한다.

- 이벤트 생성기 및 이벤트 수집기

입력 스트림에서 이벤트를 생성하여 이벤트 수집기로 전달한다. 수집된 이벤트를 이벤트 분석기로 보낸다. 본 애플리케이션에서는 실제 데이터를 받아 처리하기 어려우므로 이벤트 생성기를 이용하여 가상의 데이터를 랜덤하게 생성하여 이용한다.

- 사용자 인터페이스

사용자 GUI, 실시간 모니터링할 수 있는 뷰를 제공한다.

```
public Class userEvent{
  private String userID;
  private String userName;

  public userEvent(String userID,
    String userName){
    this.userID=userID;
    this.userName=userName;
  }
  public String getUserID(){
    return userID;
  }
  public String getuserName(){
    return userName;
  }
}
public Class SensorEvent{
  private String sensorID;
  private String sensorLocation;
  private Double temperature;
  private Double humidity;

  public SensorEvent(String sensorID,
    String sensorLoc, Double temperature,
    Double humidity){
    this.sensorID=sensorID;
    this.sensorName=sensorName;
    this.temperature=temperature;
    this.humidity=humidity;
  }
  public String getSensorID(){
    return sensorID;
  }
  public String getSensorLocation(){
    return sensorLocation;
  }
  public Double getTemperature(){
    return temperature;
  }
  public Double getHumidity(){
    return humidity;
  }
}
```

그림 9 사용자 및 센서에 대한 이벤트 설계

```
import com.espertech.esper.client.EPStatement;
import com.espertech.esper.client.EPAdministrator;
import com.espertech.esper.client.UpdateListener;

public class UserStatement
{
  private EPStatement statement;
  public UserStatement(EPAdministrator admin){
    String stmt="select count(*),userID from"
    +"UserEvent.win:time(10 sec) group by userID"
      statement = admin.createEPL(stmt);
  }
  public void addListener(UpdateListener listener){
     statement.addListener(listener);
  }
}

public class SensorStatement{
  private EPStatement statement;
  public UserStatement(EPAdministrator admin){
    String stmt="select * from UserEvent.win:"
    +"time(10 sec) group by sensorID"
    statement = admin.createEPL(stmt);
  }
  public void addListener(UpdateListener listener){
    statement.addListener(listener);
  }
}
```

그림 10 사용자 및 센서를 감시하는 리스너

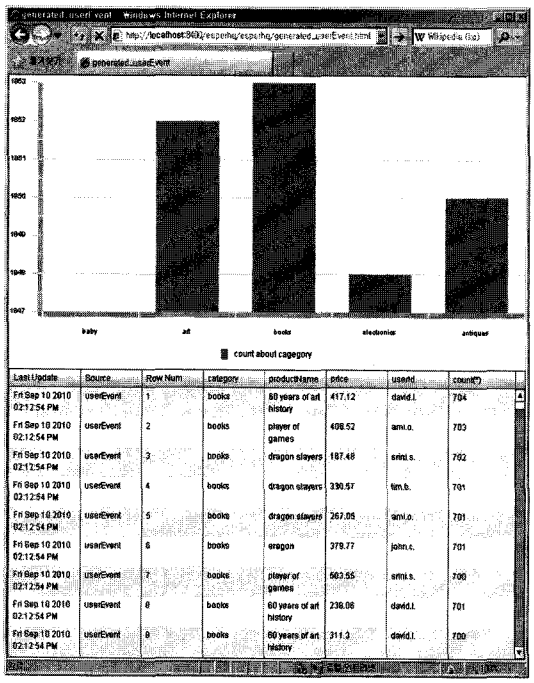
```
import com.espertech.esper.client.UpdateListener;
import com.espertech.esper.event.EventBean;

public class UserListener
    implements UpdateListener{
  public void update(EventBean[] newEvent,
                     EventBean[] oldEvent){
    System.out.println("userID="
        + newEvent[0].get("userID"));
    System.out.println("userName="
        + newEvent[0].get("userName"));
  }
}

public class SensorListener
    implements UpdateListener{
  public void update(EventBean[] newEvent,
                     EventBean[] oldEvent){
    System.out.println("sensorID="
        + newEvent[0].get("sensorID"));
    System.out.println("userName="
        + newEvent[0].get("sensorLocation"));
    System.out.println("temperature="
        + newEvent[0].get("temperature"));
    System.out.println("humidity="
        + newEvent[0].get("humidity"));
  }
}
```

그림 11 CEP 엔진 생성 및 리스너에 등록

- XML 문서 작성

사용자, 센서, RFID, 모바일 폰 등의 이벤트에 관련된 속성을 작성하여 응용프로그램에 이용한다.

그림 12는 웹의 온라인 쇼핑몰에서 상품을 구매하는 사용자들에 대한 이벤트 처리 결과를 보여준다. 카테고리는 총 5개(baby, art, books, electronics, antiques)로 분류하여 차트의 가로(x)로 표현하고 전체 이벤트의 개수는 세로(y)로 나타내었다. 그리고 5개의 카테고리 중



Last Update	Source	Row Num	category	productName	price	userId	count(*)
Fri Sep 10 2010 02:12:54 PM	userEvent	1	books	50 years of art history	417.12	david.l	704
Fri Sep 10 2010 02:12:54 PM	userEvent	2	books	player of games	408.52	ami.o.	703
Fri Sep 10 2010 02:12:54 PM	userEvent	3	books	dragon slayers	187.48	srini.s.	702
Fri Sep 10 2010 02:12:54 PM	userEvent	4	books	dragon slayers	330.57	tim.b.	701
Fri Sep 10 2010 02:12:54 PM	userEvent	5	books	dragon slayers	267.05	ami.o.	701
Fri Sep 10 2010 02:12:54 PM	userEvent	6	books	eregon	379.77	john.c.	701
Fri Sep 10 2010 02:12:54 PM	userEvent	7	books	player of games	563.55	srini.s.	700
Fri Sep 10 2010 02:12:54 PM	userEvent	8	books	50 years of art history	238.06	david.l.	701
Fri Sep 10 2010 02:12:54 PM	userEvent	9	books	50 years of art history	311.3	david.l.	700

그림 12 실행 결과

books를 구매하는 사용자들만을 추출하여 실시간으로 모니터링할 수 있도록 구현하였다.

5. 결론 및 향후과제

CEP(Complex Event Processing) 엔진을 이용하여 대용량데이터를 처리하는 기술을 제안하였다. CEP 엔진은 대용량데이터를 수집하고 이벤트들을 분석, 필터링한다. 또한 이벤트 엔진에 미리 등록해 놓은 이벤트와 새로운 이벤트를 패턴매칭하여 데이터를 추출한다. 추출된 결과를 다른 작업의 입력 이벤트로 사용하거나 요청된 이벤트에 대해 실시간으로 응답할 수 있고 유효한 데이터만 데이터베이스에 트리거할 수도 있다. 대용량 데이터를 처리하기 위한 프레임워크를 제시하고 프로토타입을 설계하고 구현하였다. 향후 과제로는 설계한 모델 및 구현한 프로그램을 이용하여 센서데이터, 모바일 폰 등의 이벤트를 복합 처리하는 기술이 필요하겠다.

6. 응용분야

응용분야로는 U-City 환경에서의 교통량 측정, 침입 및 도난방지, 환경관리, 건강관리, 자동검침, 물류(화물위치 파악 등)의 대용량데이터 처리를 위한 이벤트 처리 시스템, 센서 처리 시스템, 이벤트 기반 마켓팅, 이벤트 기반 온라인 마켓팅, 웹분석 등에 적용가능하다. 또한 최근 이슈가 되고 있는 클라우드 컴퓨팅, 차세대 전력망인 스마트 그리드의 핵심 부분인 AMI의 미터링, 실시간 모니터링이 필요한 모든 분야에 응용할 수 있다.

참 고 문 헌

[1] TTA, ICT Standardization Roadmap 2010 : Digital Contents and Software - Generation Web, pp.101-184, Telecommunication Technology Association, 2009.12.

[2] Chang Hyun Roh, "Abuse Pattern Monitoring Method based on CEP in On-line Game," The Korea Contents Association, vol.10 no.1, pp.114-121, 2010.1.

[3] Tim Bass, "Mythbusters: event stream processing versus complex event processing," In *Proceedings of the 2007 inaugural international conference on Distributed event-based systems*, vol.233, pp.1-1, 2007 systems (DEBS '07). ACM, New York, NY, USA, 1-1. 2007.

[4] Eugene Wu, Yanlei Diao, Shariq Rizvi "High-performance complex event processing over streams," *Proceedings of the 2006 ACM SIGMOD international conference on Management of data*, SESSION: Data streams, pp.407-418, 2006.

[5] Hyeonjoo Lee, Byoungju Choi, Jungwon Lee "Service Identification of Component-Based System for Service-Oriented Architecture," *The Korean Institute of Information Scientists and Engineers*, vol.35, no.2, pp.70-80, 2008.

[6] Meyoung Lee, Myoungjun Kim, "Trend of Event-Driven Service Technology," *ETRI, Analysis of Electronic Communication Trend*, vol.21, no.5, 2006.

[7] Chappell, D.A. "Enterprise Service Bus," O'Reilly, 2004.

[8] David Herreman, http://esper.codehaus.org/esper/documentation/documentation.html, Esper contributors & EsperTech Inc, 2007.

강 만 모

1998년 울산대학교 전자계산학과 학사졸업. 2000년 울산대학교 전자계산학과 석사 졸업. 2003년 울산대학교 컴퓨터정보통신공학부 박사 수료. 2006년~2009년 울산대학교 컴퓨터정보통신공학부 객원교수. 2009년~현재 울산대학교 외래강사. 관심분야는 데이터베이스, 전자상거래, 멀티에이전트, 무선 망

구 자 록

1985년 서울대학교 이학사 졸업. 1987년 서울대학교 이학석사 졸업. 1989년 서울대학교 박사 수료. 1989년~현재 울산대학교 컴퓨터정보통신공학부 부교수. 관심분야는 전자상거래, 멀티에이전트, 모델체킹

이 동 형

1996년 울산대학교 컴퓨터공학과 학사졸업. 1998년 울산대학교 컴퓨터공학과 석사 졸업. 2009년 울산대학교 컴퓨터공학과 박사 졸업. 2001년~현재 한국폴리텍VII대학 울산 캠퍼스 정보통신시스템과 부교수. 관심분야는 네트워크, 신경망, 지능형 로봇