웹 어플리케이션 응답시간 모니터링 API의 설계 및 구현

김태성° 이금석
동국대학교 컴퓨터공학과
{ruthere, kslee}@dgu.ac.kr

A Design and Implementation of Response-time Monitoring API for Web Application

Tae-Sung Kim° Keum-suk Lee
Dept. of Computer Engineering, Dongguk University

요 약

웹 어플리케이션에서 사용자의 불만이 제기되기 전에 시스템의 상태를 모니터하고 문제를 미리 파악하는 것은 서비스 제공자에게 있어서 매우 중요한 일이다. 시스템의 응답시간 측정은 시스템의 성능 문제가 발생한 부분을 찾는데 도움을 준다. 그러나 지금까지의 응답시간 측정 방법은 트랜잭션의 개념이 없거나, 웹 어플리케이션에서 사용할 수 없었다. 따라서 본 논문은 트랜잭션의 개념을 지니고, 일반 자바 프로그램 뿐만 아니라 자바 애플릿에서도 사용 가능한 응답 시간 측정 방법을 제안하였다. 논문에서 설계되고 구현된 웹 어플리케이션 모니터링(Web application monitoring:이하 WM) API를 어플리케이션에 넣음으로써 시스템은 가용성, 서비스 수준 합의, 용량 계획 등을 위해 관리 될 수 있다.

1.서론

지난 수 년 동안 인터넷 기반의 서비스, 특히 웹 서비스는 폭발적인 증가세를 보여왔다. PC week(Westel 97)의 연구에 의하면 96 퍼센트의 응답자가 서비스의 가용성을 ISP(internet service provider)를 선택하는 첫 번째 요인으로 꼽았으며, 93퍼센트의 응답자는 서비스의 성능을 두 번째 요인으로 꼽았다.[1] 그러므로 웹 서비스 제공자에 있어 서비스의 응답 시간은 사업 성공의 열쇠라고 할 수 있다.

웹 서비스의 응답 시간 측정은 시스템을 진단, 감독, 관리하는데 도움을 준다. 그러나, 예를 들어 단순히 웹 페이지의 다운로드 시간을 측정하는 방법으로는 정확한 응답 시간을 알아낼 수도 없으며, 문제되는 부분을 찾는 것은 쉽지 않다. 그 이유는 현재의 웹 서비스는 웹 클라이언트, 웹 서버, 어플리케이션 서버, 데이터베이스 서버 등 세 개 또는 네 개의 계층으로 이루어진 경우가 많기 때문이다.

본 논문에서는 웹 어플리케이션의 응답시간을 측정할 수 있는 모니터링 기법을 제안한다. 모니터링을 통해 어플리케이션이 잘 작동하는지, 트랜잭션이 완벽히 끝나는지, 병목현상이 일어나는지 등에 대한 질문에 답할 수 있다. 특히 웹 어플리케이션에서 웹 클라이언트로 많이 쓰이는 자바 애플릿(Applet) 내에서의 응답 시간을 측정할 수 있는 방법을 제안하고 구현하였다.

본 논문의 2장에서는 웹 어플리케이션 모니터링의 관련연구를 살펴보고, 3장에서는 WM API에 대한 자세한 설명, 4장에서는 구현, 5장에서는 결론 및 향후연구를 살펴본다.

2 관련연구

2.1 기존의 모니터링 방법

Hewlett Packard의 인터넷 서비스 모니터링 프로그램인 Firehunter[1]는 서비스 측정을 크게 능동적 측정과 수동적 측정의 두 가지로 나누었다. 능동적 측정이란 가상의 클라이언트를 두어서 실제 트래픽을 만들어 이를 측정함을 말한다. 수동적 측정은 서버가 모니터링의 능력을 가지고 실제 클라이언트의 응답 시간을 모니터링하는 것이다. Firehunter는 이런 두가지 방식의 서비스 측정 방식을 적절히 섞어 사용하였다. 하지만, 이런 방식의 문제점은 트랙잭션의 개념이 없는데 있다. 현재의 웹은 정적인 페이지가 아닌 세 개 또는 네 개의 계층을 통해 만들어진 동적인 페이지이기 때문에 각 계층의 트랜잭션을 측정해야 정확히 문제 되는 부분을 찾아 낼 수 있다.

ETE[2]에서는 트랜잭션의 개념을 도입했고 개발이 끝난 후 서비스 개시 전이나 혹은 서비스 중간에 트랜잭션의 정의를 할 수 있도록 했다. 개발시에 트랜잭션을 정의할만한 곳에 이벤트를 적고 이렇게 쓰여진 이벤트는 모두 디렉터라는 모니터링 데몬(monitoring daemon)에게로 전달된다. 디렉터는 서비스 제공자가 정의한 트랜잭션의 정의를 담고 있어서, 일련의 이벤트를 살펴 보고 매치되는 트랜잭션을 찾아낸다. 이러한 융통성은 그 만큼 큰 비용을 수반한다. 트랜잭션을 동적으로 정의하고 가능한 모든 이벤트를 디렉터로 보내는 일은 CPU와 네트웍 대역폭에 많은 부하를 줄 수 있다. 개발자가 만들어 내는 이벤트를 통해 서비스 제공자가 트랜잭션을 정의하려면, 그 만큼 개발자는 가능한 모든 이벤트와 이에 대한 상세한 설명을 덧 붙여야 한다. 이는 오히려 개발자가 트랜잭션을 정의하는 것보다 더 많은 시간과 노력을 요구하고, 또 경우에 따라서는 개발자가 의도 하지 않은 방향으로 트랜잭션이 정의될 수 있다.

CMG(Computer Measurement Group)의 ARM[3]은 쓰기 쉬운 API를 통해 개발 시점에 트랜잭션을 정의할 수 있는 방법이다.

개발자는 트랜잭션의 시작과 끝을 간단한 API를 통해 시스템에 모니터링 능력을 부여 할 수 있다. 연관된 트랜잭션은 부모와 자식 관계로 표현할 수도 있고, 응답 시간 뿐만 아니라 서비스 내의 상태를 모니터링 할 수 있다. 그러나 ARM의 API는 C 언어에서만 사용 할 수 있다(버전 2.0). 버전 3.0(draft)은 자바에서 쓸 수 있는 API를 만들 예정이다. 하지만 버전 3.0에서도 다운로드된 호스트 외에는 다른 호스트와는 네트웍 연결을 할 수 없다는 제한 때문에 애플릿에서는 API를 사용할 수 없다. 또 전체 시스템이 아닌 시스템의 일부만을 모니터링하는 기능이 없다.

3.WM(web-application monitoring)설계

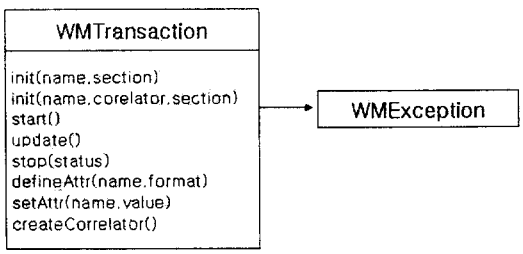
WM은 자바로 쓰여진 웹 어플리케이션을 모니터링 한다. 애플릿을 포함한 모든 종류의 자바로 쓰여진 프로그램을 모니터링 할 수 있다.

3.1 WM API

WM API 를 사용하기 앞서 가장 중요한 단계는 비즈니스 트랜잭션을 정의하는 것이다. 개발자는 어떤 데이터를 누가 필요로 할지 생각해 보아야 한다. 이 과정은 시스템 관리자나 서비스 제공자와 의논을 필요로 할 수 도 있다. 트랜잭션을 정의하는 일반적인 방법은 우선 사용자에게 보여지는 부분이나, 주요 비즈니스 오퍼레이션을 트랜잭션으로 정하고, 이 트랜잭션에 의존하는 데이터베이스 연동, RPC(remote procedure call), 네트웍 연결 등을 트랜잭션으로 정의한다. 이들 트랜잭션은 부모-자식의 관계를 가지게 된다. 이들은 문제를 분석하고 프로그램을 고치고 시스템과 네트웍 설정을 바꾸는데 가치있는 데이터가 된다.

트랜잭션을 정의한 후에는 프로그램에 WM API를 부르는 코드를 삽입하는 것이다. WM API는 쉽고 간단하므로 이 과정은 그다지 시간과 노력을 필요치 않는다. 그림 1은 WM API의 클래스 다이어그램이고 각 API의 기능은 다음과 같다.

init(name,section) : 트랜잭션을 초기화하기 위한 것이다. name은 트랜잭션의 이름이고 section은 트랜잭션이 속한 영역이다. 영역의 기본값은 WMTransaction.DEFAULTSECTION이다. 이 함수로 트랜잭션은 시스템에서 유일한 아이디를 부여받는다. 이는 이름과는 다른 것으로써 이름은 트랜잭션의 종류를 의미할 경우가 많다.



[그림 1] WM API의 클래스 다이어그램.

init(name,corelator,section) : 이 함수는 트랜잭션이 부모-자식의 관계를 가질 때, 자식 트랜잭션의 함수이다. 부모 트랜잭션을 통해 만들어진 관계자(correlator)를 통해 부모-자식을 표현한다.

start() : 트랜잭션의 시작을 모니터링 데몬에게 알리는 함수이다. 이 함수를 트랜잭션의 시작에 삽입해야 한다.

update() : 이 함수는 트랜잭션이 진행중임을 모니터링 데몬에게 알리는 것이다. 일정 시간이 지나면 모니터링 데몬은 시작되었으나 끝나지 않은 트랜잭션을 실패로 기록한다.

stop(status) : 트랜잭션이 끝났음을 알리는 함수이다. status는 성공한 경우에는 WMTransaction.SUCCEED를 실패한 경우에는 WMTransaction.FAILD 값을 갖는다.

defineAttr(name,format) : 추가된 속성을 정의하는 함수이다. format의 종류는 3장 2절에 설명되어 있다.

setAttr(name,value) : 추가된 속성에 실제 값을 넣는 함수이다.

createCorrelator() : 관계자를 생성하는 함수이다. 부모 트랜잭션에서 관계자를 생성해서 자식 트랜잭션의 init함수에 전해줌으로써 부모-자식의 관계를 표현한다.

3.2 추가된 속성

WM API는 응답 시간 측정 뿐만 아니라 어플리케이션 내의 다른 데이터를 모니터링 데몬으로 보낼 수 있는 방법을 제공한다. 이를 추가된 속성이라 부르고 추가된 속성을 위한 함수는 defineAttr, setAttr이다. 예를 들어 어플리케이션 내의 특정 버퍼의 크기, 사용자 이름, 에러 설명 등을 모니터링 데몬으로 보낼 수 있다. 추가된 속성으로 보낼 수 있는 데이터의 형식은 다음과 같다.

WMTransaction.GAUAGE32, WMTransaction.GAUAGE64,
WMTransaction.COUNT32, WMTransaction.COUNT64,
WMTransaction.NUMERIC32, WMTransaction.NUMERIC64,
WMTransaction.STRING32, WMTransaction.STRING64 .

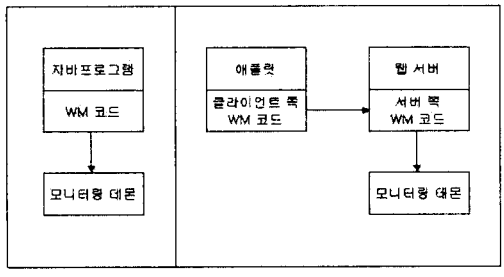
3.3 연관된 트랙잭션과 영역.

트랜잭션은 부모-자식간의 관계를 가질 수 있다. 이는 자식 트랜잭션의 init함수에서 두 번째 매개변수에 부모 트랜잭션의 createCorrelator함수에서 반환된 값을 넣음으로써 표현한다. 모니터링 데몬은 이를 수집해 부모-자식을 내부 표현으로 정리한다.

어플리케이션의 일부만을 모니터링 할 수 있는데 이는 init 함수의 마지막 매개변수인 영역을 사용해서 가능하다. 예를 들어 새로 추가된 웹 서비스 만을 모니터링하고 싶을 때는 그 웹 서비스안의 트랜잭션을 같은 영역으로 지정하는 것이다.

3.4 애플릿에 대한 사항

애플릿은 다운로드된 호스트 외에는 네트웍 연결을 만들지 못하는 제한이 있다. [5]에 의하면 이런 제약 사항 때문에 일반 자바 프로그램에서 사용하는 WM API와 애플릿에서 사용하는 API는 다를 수 밖에 없다고 언급하고 있다. 모니터링 데몬에게 모니터링 정보를 보내는 코드가 서버와 클라이언트 부분으로 나뉘어져야 한다. 그림 2는 일반 자바 프로그램에서 WM API를 불렀을 때와 애플릿에서 불렀을 때의 개념도이다.



[그림 2] 자바 프로그램과 애플릿에서 WM API의 차이

애플릿에서 WM API를 부르려면 다른 클래스를 사용해야 한다. WMTransactoion과 WMException 대신에 AWMTransaction 과 AWMException 를 사용하며 함수는 WMTransacton과 똑같다.

3.5 관리 기능

모니터링 데몬을 통해 모니터링되는 영역을 정할 수 있다. 모니터링 영역은 더 이상 모니터링 할 필요가 없는 영역을 지정하거나 이미 사용자에게 배포해서 코드를 수정할 수 없을 때 유용하다. 세밀한 영역 사용은 모니터링 때문에 생기는 부하를 크게 줄인다. 데몬은 시작했으나 일정 시간 동안 끝나지 않은 트랜잭션을 실패로 처리한다. 시스템, 어플리케이션 종류에 따라 일정 시간은 변화가 크므로 이 일정시간을 조정할 수

있는 기능을 가지고 있다. 그 외에도 데몬은 관리를 위한 세 가지 기능을 가지고 있다.

① **알람 기능** : 트랜잭션의 응답시간이 어느 이상 커지면 관리자에게 알람하는 기능이다.

② **수집 기능** : 일정 시간 예를 들어, 15분마다 트랜잭션의 응답시간 평균과 표준편차를 보고하는 기능이다.

③ **추적 기능** : 문제 발생시 이를 자세히 추적하기 위해 부모 트랜잭션 뿐만 아니라 자식 트랜잭션과 그 트랜잭션의 추가된 속성까지 보여주는 기능이다.

3.6 코드 예제

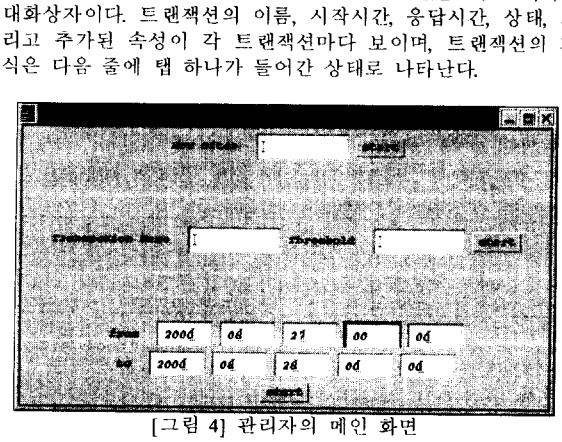
그림 3을 보면 tran1은 start 함수 호출이전에 defineAttr 함수로 추가된 속성을 정의하였다. 추가된 속성의 값을 보내는 setAttr함수는 stop함수 이전에 몇번이고 호출 할 수 있다. 반복문안의 tran2의 init 함수에서 두번째 매개변수로 tran1의 관계자를 보낸다. 이로써 tran2는 tran1의 자식의 관계를 가진다. 반복문의 끝부분에서 tran1의 update함수 호출은 모니터링 데몬에게 트랜잭션 tran1이 계속 진행 중임을 알리는 것이다. 코드의 마지막 부분에서 tran1은 tranNew라는 새로운 트랜잭션으로 다시 초기화 되고 있다.

```
try{
    WMTransaction tran1 = new WMTransaction();
    WMTransaction tran2 = new WMTransaction();
    tran1.init("tran1",WMTransaction.DEFAULTSECTION);
    tran1.defineAttr("queueSize",WMTransaction.GAUAGE32);
          tran1.start();
    ...
    tran1.setAttr("queueSize",19);
     for(int i = 0 < i < size; i++){
       tran2.init("tran2",tran1.createCorelator(),1);
        tran2.start();
        ...
       tran2.stop(WMTransaction.SUCCEED);
        tran1.update();
     }
    tran1.stop(WMTransaction.FAIL);
    ...
    tran1.init("tranNew",WMTransaction.DEFAULTSECTION);
    tran1.start();
    tran1.stop(WMTransaction.SUCCEED);
} catch(WMException e){
    System.out.println(e.getMessage()) }
```

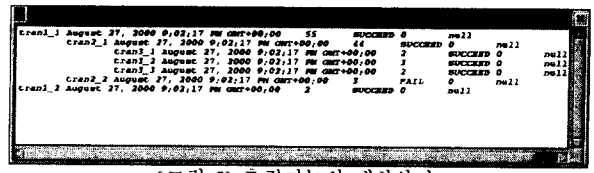
[그림 3] WM API 사용 예제.

4.구현 및 실험

그림 4는 관리자의 메인 화면이다. 맨 위의 패널은 수집기능을,가운데 패널은 알람기능을, 아랫 쪽 패널은 추적기능을 위한 패널이다. 그림 5는 추적기능을 실행 했을 때 보여지는 대화상자이다. 트랜잭션의 이름, 시작시간, 응답시간, 상태, 그리고 추가된 속성이 각 트랜잭션마다 보이며, 트랜잭션의 자식은 다음 줄에 탭 하나가 들어간 상태로 나타난다.

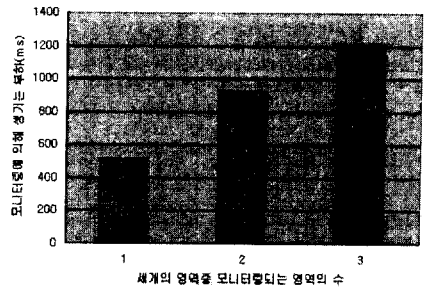


[그림 4] 관리자의 메인 화면



[그림 5] 추적기능의 대화상자.

그림 6은 영역 사용으로 줄일 수 있는 모니터링 부하를 보인 것이다. 그래프는 세개의 영역이 설정 되었을 때, 모니터링 되는 영역의 수에 따라 모니터링에 의한 부하를 나타낸 것이다. 전체를 모니터링하기 보다는 필요한 부분만 모니터링함으로써 모니터링에 의한 부하를 줄일 수 있음을 알 수 있다. 두개의 영역에 의한 부하가 세개의 영역에 의한 부하의 2/3를 넘는 이유는 init함수를 호출한 이후에 모니터링 영역이 아님을 알기 때문이다. 모니터링 영역 아닌 것을 알고 나면 init이후의 함수 호출은 무시된다.



[그림 6] 세개의 영역중 모니터링되는 영역의 수에 따라 생기는 모니터링에 의한 부하.

5.결론 및 향후 연구

응답 시간 측정으로 시스템은 가용성, 서비스 수준 합의,용량 계획 등을 위해 관리 될 수 있다. 본 논문은 일반 자바 프로그램뿐만 아니라 웹 클라이언트로 자주 쓰이는 애플릿에서도 사용 가능한 응답 시간 측정 방법을 제안하고 구현하였다. 또한 영역 사용을 통해서 모니터링 부하를 줄이는 방법도 제안하였다.

응답 시간 측정은 분산 시스템 관리에서 성능 측정에 의한 관리 방법에 속한다. 향후 CIM(Common Information Model)이나 JMX(Java Management Extension)같은 관리 표준에 맞추어 만드는 것이 필요하다. 자바 뿐만이 아닌 웹 프로그램에 많이 쓰이는 다른 스크립트 언어에서도 사용 가능하도록 보완 해야 겠다.

6.참고문헌

[1]C.Darst, S. Ramanathan "Measurement and management of internet services" Integrated Network Management, 1999

[2]Josep L. Hellerstein, Mark M. Maccabee, W. Nathaniel Mills and John J. Turek "ETE: A Customizable approach to measuring end-to-end response time their components in distributed systems ", 19th IEEE International Conference on Distributed Computing systems , 1999

[3] Mark W. Johnson "Monitoring and diagnosing application response time with ARM", IEEE Third International Workshop on Systems Management,1998

[4]Sun Microsystems "Java Management Extensions instrumentation and agent specification ", http://java.sun.com

[5]Mark W. Johnson "Measuring service level of Java Applets E-business applications" CMG98 Proceedings.

[6]CMG "Application Response Measurement user 's guide", http://www.cmg.org