규칙기반 웹 서버 장애 진단 추론 기법

윤정미, 한정수, 정진욱
성균관대학교 전기전자 및 컴퓨터공학부
e-mail : jmyun@songgang.skku.ac.kr

Rule-based Reasoning Scheme for Web Server Fault Diagnosis

Jungmee Yun, JeungSoo Han, Jin-Wook Chung
School of Electrical and Computer Engineering, Sungkyunkwan University

요 약

이 논문에서는 웹 서버에 발생할 수 있는 장애 항목들을 정의하고, 발생한 장애를 자동적으로 진단하고 이를 검출하기 위한 방법을 규칙기반 추론기법을 사용하여 제안하였다. 즉, 웹 서버 관리에서 발생할 수 있는 장애 항목을 정의하고, 장애를 진단하기 위한 규칙을 제안하였는데, 장애 항목으로는 프로세스 장애, 서버 과부하, 인터페이스 장애, 구성 및 성능 장애를 정의하였으며, 각 장애 항목을 진단하기 위한 지식을 활성 네트워크기법을 적용하여 표현하고, 이를 시스템 레벨 장애 진단 생성규칙과 서비스 레벨 장애 진단 생성규칙으로 정형화하였다. 그리고 제안한 장애 진단규칙의 타당성을 증명하기 위한 장애 환경 구성을 구성하고, 각 장애 환경에 대한 생성규칙 적용과정을 실험을 통하여 제시하였다. 이 논문에서는 기하 급수적으로 증가하는 웹 서버의 장애를 관리하기 위한 메커니즘 제안함으로써 웹 서버 관리에 소요되는 관리자의 노력을 최소화할 수 있는 지능적인 장애 관리를 위한 방법론을 제시하고자 한다.

1. 서론

1992 년 웹 서비스가 첫 선을 보인 이후 2 년 만에 웹 서비스가 인터넷을 독점했다고 할 정도로 사용자들 사이에서 돌풍을 일으키기 시작하였다[1]. 웹이 인터넷에 존재하는 많은 소프트웨어 가운데 하나이며, 여기에 사용되는 HTTP 프로토콜도 TCP/IP 프로토콜 스택의 응용계층에 해당하는 프로토콜에 지나지 않을 뿐인데도 불구하고 이제는 웹이 곧바로 인터넷을 의미할 정도로 그 사용이 급격히 증가하고 있다. 이것이 가능하게 된 이유로는 여러 가지를 꼽을 수 있으나 크게 손쉽게 사용할 수 있다는 것과 웹 서버의 설치가 간단하다는 것을 들 수 있다. 그 결과 국내 현존하는 웹 서버의 개수만도 십 만개 정도에 이르나[2], 웹 서버의 증가와 비교하여 장애에 대한 적절한 대처가 이루어지고 있지는 않다. 이와 관련하여 웹 서비스 신뢰성을 향상시키기 위하여 웹 서버 성능 튜닝 기반의 다양한 방법론이 제안되고 있으나, 대부분의 연구가 웹 서버 자체의 성능 평가나 SNMP 를 기반으로 한 자원 관리 중심으로 이루어지고 있으며[3][4][5], 부분적으로 응용 서비스의 트래픽 패턴 분석을 통한 성능관리 및 장애 관리방법론에 대한 접근이 이루어지고 있는 실정이다.[6] 그러나 현재의 접근 방법은 웹 서버의 장애 관리에 대한 접근은 거의 이루어지지 않고 있으며, 단순히 웹 서버의 성능 향상 위주의 관리만 다루어지고 있는 실정이다.

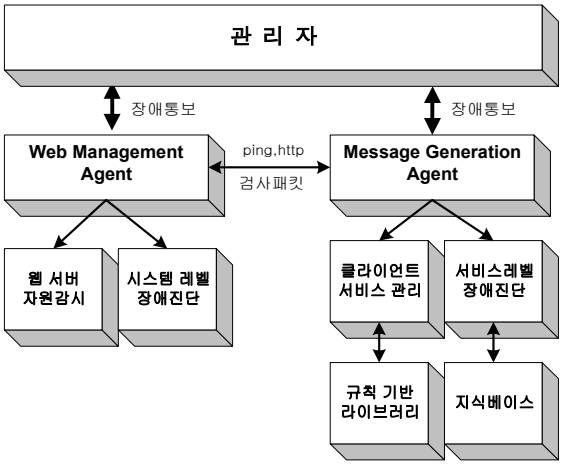
따라서 이 논문에서는 응용 서비스, 특히 응용 트래픽의 80%정도를 점유하는 웹 서비스 장애 관리의 중요성을 인식하고 장애 관리를 체계적으로 수행하기 위한 장애 검출 규칙 정형화 기법을 제안하고자 한다. 특히, 장애관리 지식을 정형화된 형태로 표현함으로써 체계적이며 지능적인 장애 검출관리방법을 제시한다. 또한 실험을 통한 장애환경 조성 및 장애 규칙 진단을 통하여 실제 환경에서의 장애 진단 규칙의 적합성에 대해서 살펴보도록 하겠다.

2. 시스템 구조 및 장애 항목정의

2.1 시스템 구조

웹 서버 장애 관리 시스템은 웹 서비스를 사용하는데 있어서 발생할 수 있는 장애를 검출하고 이와 관련된

임계치를 적응적으로 변화시키는 시스템으로 전문가 지식을 체계화된 진단 규칙으로 표현하고, 실측 데이터와 장애 발생 파라미터의 관계추론을 통하여 적응적으로 장애를 추론한다. 본 논문에서는 이를 위한 시스템 구조로 아래 (그림 1)과 같은 시스템 구조를 정의하였다. 즉, 웹 서버가 설치되어 있는 시스템 내에 위치하여 웹 서버의 상태를 감시하는 Web Management Agent(WMA)와 주기적으로 웹 서버로 서비스 요청 패킷을 보내고 이 검사 패킷에 대한 서버의 응답을 이용하여 서버의 상태를 진단하는 Message Generation Agent(MGA)로 이루어진다.



(그림 1) 웹 서버 장애 관리 시스템의 구조

검사 패킷에 대한 응답으로 웹 서버는 HTTP 버전, 상태코드, 응답 구문으로 이루어진 상태라인을 포함하는 응답 메시지를 보내게 되며, 이때의 상태코드를 이용하여 서버의 상태를 진단하게 되는데, 서버가 구동 가능한 상황에서 서버 장애가 발생하는 경우 에러 코드를 상태코드에 추가하여 보내게 되는데, 이 응답 메시지를 분석을 통하여 MGA 시스템에서의 장애 진단 추론 과정이 활성화된다.

2.2 웹 서버 장애 항목

응용서비스 분야에 있어서의 장애란 물리적인 장애뿐만이 아니라 사용자의 관점에서의 장애까지를 포함하는 것으로 서비스 제공, 응답시간 등의 사용자 QoS를 지원하지 못하는 경우도 장애가 발생한 것이라 할 수 있다.[7]

<표 1> 웹 서버의 장애 항목

장애 항목	발생 증상
프로세스 장애	프로세스 다운
	하부 프로세스 급증
서버 과부하	서비스 폭주
	시스템 과부하
	응답 지연시간 증가
	HTTP 트래픽 급증
인터페이스 장애	라우팅 구성장애
	인터페이스 다운
구성정보 오설정	서버 구동 불가
	성능 저하

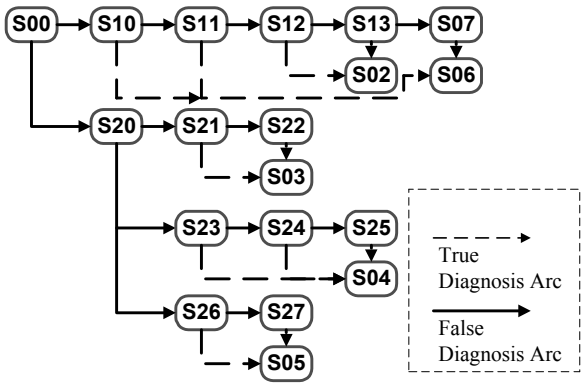
<표 1>에서 보는 바와 같이 웹 서버의 장애 항목은 크게 프로세스 장애, 서버 과부하, 인터페이스 장애, 구성정보 오설정으로 정의할 수 있으며, 각 장애 항목별로 발생하는 진단 증상을 도출할 수 있다. 첫번째 장애 항목인 프로세스 장애의 경우 이와 수반하여 발생될 수 있는 증상들로 프로세스 구동이 안 되는 경우와 하부 프로세스의 폭주가 발생하는 상황들을 들 수 있으며, 이러한 장애는 성능과 관련된 구성정보의 오 설정으로 인한 원인들에 기인하여 발생될 수 있다. 다음으로 서버 과부하는 웹 서버 시스템 상태가 과부하가 되거나 서비스를 제공하는 서버 프로세스수의 폭주, 서비스 제공 시간의 증가 등의 장애 상황이 발생하는 것을 말하는 것으로 이 경우 웹 서버에서는 서버 장애 코드 중 Service Unavailable를 이용하여 서버의 장애를 알리게 된다. 관련하여 발생할 수 있는 장애 증상으로는 시스템의 CPU 사용량 증가, 응답 시간 증가, HTTP 트래픽의 급증 등이 있을 수 있다. 세 번째로 인터페이스 장애는 내부적으로 웹 서버의 장애가 없는데도 불구하고 서비스를 제공하지 못하는 것으로 논리적으로 시스템의 인터페이스가 다운되거나 서버의 네트워크 오설정이 그 원인이 된다. 마지막 구성정보 오설정은 올바르지 못한 구성환경 설정으로 인하여 웹 서버의 성능 저하가 발생할 수 있으며 이는 장애를 초래하는 원인으로 작용할 수 있다.

3 장애진단규칙

웹 서버의 장애 발생은 고정적인 요인보다는 다양한 환경 요인들에 의해 발생한다. 영향을 미치는 환경 요인으로는 시스템 자원 가용성, 서버 평균 부하, 서비스 요청 횟수 등 다양한 원인이 있으며 이러한 원인들이 복합적으로 작용하여 장애가 발생한다. 그러므로 응용 서비스 장애 관리를 하기 위해서는 정형화된 형태의 관리 기법이 필요하다. 본 논문에서는 이러한 웹 서버 장애 검출 지식을 체계화하기 위하여 규칙 기반추론(RBR: Rule - Based Reasoning)기법을 사용하였으며, 이때 사용된 규칙 표현 기법은 절차적 규칙을 토대로 하였다.

여기서, 선언적 규칙이라고도 불리는 지식규칙은 문제영역에 대한 모든 사실과 관계들을 그대로 기술하는 방식을 말하며, 절차적 규칙이라고도 불리는 추론규칙은 이미 알려져 있는 사실이나 관계 및 규칙들을 토대로 문제를 해결하기 위한 새로운 규칙을 생성하는 과정을 말한다. 이러한 진단 규칙표현을 위해서는 달성 목표의 정의, 활성 네트워크(Active Network)표현, 규칙정의 과정이 필요하다. (그림 2)가 장애관리를 위한 활성 네트워크를 표현한 것으로 달성목표집합으로 {S01, S02, S03, S04, S05, S06, S07}이 해당

되며, 초기 진단 노드인 S00 노드로부터 조건에 따라 활성 네트워크의 노드를 방문하게 되며 달성목표집합에 도달 시 진단과정의 진단이 완료되게 된다.



(그림 2) 장애 검출 활성네트워크

각 상태 노드들에 대한 정의는 <표 2>와 같으며 각 상태들의 이동은 장애 진단규칙에 의하여 정의되어진다. 표에서의 α, β, γ 는 수집 데이터에 의해서 설정되는 임계치를 나타내며, 상태 집합은 장애 항목을 표현하는 상태집합({S02, S03, S04, S05, S06, S07})과 장애 진단 과정을 표현하는 상태집합({S00, S01, S10, S11, S12, S13, S20, S21, S22, S23, S24, S25, S26, S27})으로 나누어진다. 탐색 경로는 규칙 결정 파라미터에 의한 상태 값이 FALSE 값을 가질 때는 False Diagnosis Arc 를 따라서 이동하고, TRUE 값을 가질 경우에는 True Diagnosis Arc 를 따라 이동하게 된다.

<표 2> 상태 정의

상태	정의	상태	정의
S00	Generation test packet	S12	Non exist http daemon
S01	No failure	S13	No response(ICMP)
S02	Network failure between MGA and WS	S20	Include error code in status line
S03	Internal server error	S21	Include configuration error
S04	Server overload	S22	Contain CGI error in log file
S05	DNS failure	S23	Excess maximum process
S06	Inoperable http daemon	S24	Resource usage of http daemon process >β
S07	Process fault	S25	Resource usage of entire system process > γ
S10	No response(HTTP)	S26	Abnormal state of DNS resolution
S11	Response time > α	S27	Abnormal state of forwarding DNS

<표 3> 은 시스템 레벨에서의 장애 발생 여부를 판별하기 위한 시스템 레벨 장애 진단 생성규칙은 사건 *P1* 에 의해서 규칙 *R1* 이 활성화되며, 이 진단 규칙에 따라 어떠한 장애가 발생하였는지를 파악하게 된다.

<표 3> 시스템 레벨 장애 진단 규칙

진단규칙	*R1*
사건	*P1* **P1:no response from web server*
활성 네트워크	*{S07, S10, S11, S12, S13}*
생성 규칙	*Key:* L: *Case Library = {P,P2,P3}* S: *Current State* A: *Current Active Network* *= {,S07,S10,S11,S12,S13}* $S_{initial}$ *:Initial State =* $\{S_{00}\}$ NMBANCA(A,L,S) = WHILE *not(stoping_criterion_satisfied(S))* *a = retrieve_case(A,S,L)* DO($1 \le i \le$ *rule_length(A)*) $S_{previous}$ *:= S* *S:=execute(action(a,i),*$S_{previous}$ *)* IF(*failure*(*S*)) THEN *RETURN*{*A,L*} IF(S_{07}) {*A,L*}:=*learn*(*S*,$S_{initial}$,*a,i,A,L*) *RETURN* {*A,L*}

<표 4>의 규칙 *R2* 는 웹 서버의 구성 및 성능장애를 진단하는 서비스 레벨 장애 진단 생성규칙으로 상태 코드 분석을 통하여 사건 *P2,P3* 가 발생한 경우 활성화되며, 이 진단 규칙에 따라 어떠한 장애가 발생하였는지를 파악하게 된다.

<표 4> 서비스 레벨 장애 진단 규칙

진단규칙	*R2*
사건	*P2/P3* **P2: receive response message with error status code* **P3 : excess response time*
활성 네트워크	*{S20, S21, S22, S23, S24, S25, S26, S27}*
생성 규칙	*Key:* L: *Case Library = {P,P2,P3}* S: *Current State* A: *Active Network* *=* $\{S_{20},S_{21},S_{22},S_{23},S_{24},S_{25},S_{26},S_{27}\}$ $S_{initial}$ *:Initial State =* $\{S_{00}\}$ NMBANCA(A,L,S) = WHILE *not(stoping_criterion_satisfied(S))* IF($Pi \in \{S_{21}\}$) THEN $A = \{S_{21},S_{22}\}$ IF($Pi \in \{S23\}$) THEN $A = \{S_{23},S_{24},S_{25}\}$ IF($Pi \in \{S_{26}\}$) THEN $A = \{S_{26},S_{22}\}$ *a = retrieve_case(A,S,L)* DO($1 \le i \le$ *rule_length(A)*) $S_{previous}$:= S S := *execute(action(a,i),*$S_{previous}$ *)* IF(failure(*S*)) THEN *RETURN{A,L}* IF($S \in \{S_{24},S_{25},S_{23}\}$) {*A,L*}:=*learn(S,* $S_{initial}$*, a, i, A,L)* *RETURN {A,L}*

<표 3>과 <표 4>는 INBANCA 기법에서 정의하고 있는

절차적 규칙 표현방식에 따라 기술한 생성규칙으로 추론과정은 진단할 활성 네트워크의 범위를 줄여나가면서 최종 목적 노드에 도달하게 된다. 이 때 수집 데이터의 임계치는 지식베이스에 저장이 되게 되며, 과거의 데이터 중 가장 적합한 값을 추출하여 사용하며, 만약 규칙을 만족하지 않는 경우 절차적 규칙에 따라 새로운 규칙을 생성하게 된다.

4. 실험 및 고찰

제시한 웹 서버 장애 진단 모델을 실제 환경에 적용시켜 장애를 감지하기위해 제안한 규칙의 타당성 여부를 살펴보도록 하겠다. 웹 서버의 장애를 검출하기 위해서 본 실험에서 사용된 환경은 <표 5>에서 보는 바와 같이 1 대의 서버와 3 대의 클라이언트로 구성하였다. 서버 환경은 현재 웹 서버의 60%정도를 차지하는 NCSA 웹 서버를 사용하였으며[8], 운영체제는 리눅스를 기반으로 하여 실험하였다.

<표 5> 실험 대상 시스템

Server Machine		O/S	Memory
S1	Pentium II 300	Linux RedHat5.0	64MB
Client Machine		O/S	Memory
C1	Sparc	SunOS 5.5.1	64MB
C2	Sparc	SunOS 5.6	64MB
C3	Sparc	SunOS 5.6	128MB

<표 5>의 대상 시스템을 기반으로 장애 환경을 구성하지 위해 Webstone(SGI)을 이용하였는데, WebStone 은 하드웨어 구조나 운영 체제등과 무관하게 웹 서버의 성능을 평가하는 벤치마크로 본 실험에서 사용하기에 적절하다고 할 수 있다.
즉, WebStone 을 C1,C2,C3 클라이언트에 설치하고 클라이언트별로 HTTP /1.0 GET 요청을 이용하여 작업부하를 점진적으로 증가시킴으로써 장애 환경을 구성하였으며, <표 6>와 같은 실험결과를 얻을 수 있었다. 표에서 알 수 있듯이 의도한 장애 항목을 규칙에 의거하여 검출할 수 있었으며, 이때 사용된 임계치는 시스템 감시를 통하여 적응적으로 설정됨을 확인할 수 있었다.

<표 6> 실험결과

Test	Fault	Threshold	Rule	Result
T1	S04	β > 80(%)	R2	success
T2	S06	α >200(ms)	R2	success
T3	S03	Configuration modify	R1	success

5. 결론

이 논문에서는 가변적인 웹 서버 환경에 적응할 수 있는 장애관리 메커니즘을 규칙기반 추론 기법을 사용하여 정형화된 형식의 진단 규칙으로 정의하였다. 규칙 표현 기법과정에 따른 달성 목표의 정의, 서버 상태감시 및 장애 진단 지식의 활성 네트워크(Active Network)표현, 절차적 규칙을 정형화하였다. 또한 진단 규칙을 웹 서버의 자원 및 구성을 주기적으로 감시하는 시스템 레벨 장애 진단 생성규칙과 검사 패킷을 이용한 성능 및 서비스 장애를 진단하는 서비스 레벨 장애 진단 생성규칙으로 계층화하여 계층적인 장애관리가 가능하게 하였다.
이와 같은 웹 서버 장애 관리를 위해 시스템 레벨 장애 추론을 수행하는 WMA 와 서비스 레벨 장애 추론을 수행하는 MGA 로 시스템을 구성하였으며, 추론 엔진과 규칙/지식베이스를 이용하여 지능적인 장애 진단을 수행한다. 그리고 웹 성능 벤치마크인 Webstone 을 이용한 장애 환경 구성 및 각 장애 항목에 대한 장애 진단 생성규칙 적용 결과를 보임으로써 제안한 장애 진단규칙 메커니즘의 타당성을 실험을 통하여 증명하였다. 이러한 웹 서버 장애 관리 기법은 급속히 증가하는 웹 서버의 관리를 위한 관리자의 노력과 비용을 줄이고, 좀더 적응적이며 지능적인 웹 서비스 장애관리가 가능하게 할 것이다.

6. 참고문헌

[1] Martin F.Arlitt and Carey L. Williamson, " Web server workload characterization", in Proceedings of the ACM SIGMETRICS Conference on Measurement & Modeling of Computer Systems, 1996
[2] Paul E. Hoffman, "Web Wervers Survey", http://www.webcompare.com/web_server.html, 1999
[3] 한정수, 안성진, 정진욱 " Web-based performance manager system for a Web server", Network Operations and Management Symposium NOMS 98., IEEE,1998
[4] 홍원택, 최영수, 정진욱, "효율적인 성능관리를 위한 TCP/IP 네트워크 이용률 예측에 관한 연구", 한국정보처리학회 추계학술발표논문집, 1998
[5] Yiming Hu, Nanda, A, Qing Yang, "Measurement, analysis and performance improvement of the Apache Web server", Performance, Computing and Communications Conference IEEE International , 1999
[6] Goldszmidt, G.S," Load management for scaling up Internet services", Network Operations and Management Symposium NOMS 98., IEEE,1998
[7] Polze, A, Richling. J, Schwarz, J, Malek, M," Towards predictable CORBA-based Web-services", Object-Oriented Real-Time Distributed Computing (ISORC '99). Proceedings. 2nd IEEE International Symposium, 1999
[8] Lee KangChan, "국내 WWW 서버 조사(WWW Server Survey in Korea)", URL: http://sharon.comeng.chungnam.ac.kr/~dolphin/Server/compare.html