유비쿼터스 환경에서의 시스템 신뢰성을 위한 상황 인식형 자가 치유 시스템

이미선, 박정민°, 이은석
성균관 대학교 컴퓨터 공학과
msviolet97@selab.skku.ac.kr, jmpark°@selab.skku.ac.kr, eslee@ece.skku.ac.kr

A Context-Awareness Self-healng System for System Reliability in Ubiquitous Computing Environment

Miseon Lee, EunSoek Lee
School of Information and Communication Engineering, Sungkyunkwan university

요 약

오늘날 대부분의 분산 컴퓨팅 환경들은 매우 복잡하다. 이러한 분산 환경위에 동작하는 시스템들은 문제 발생시 원인의 진단과 치유에 소비되는 시간이 높다. 이에 유비쿼터스환경으로 발전하기 위하여 네트워크를 구성하는 시스템들의 문제나 장애의 자가 진단 및 치유에 대한 요구가 높아지고 있다.

따라서 본 논문에서는 기존의 자가 치유 시스템에서 연구되어왔던 일률적인 치유가 아닌 각각의 시스템의 상황을 인식하고 그것에 맞추어 치유전략을 달리하는 상황인식 기능과 치유 결과 분석을 통해 치유전략을 자발적으로 개선해 나가는 기능을 제공한다. 제안 시스템은 시뮬레이션을 통하여 그 유효성을 입증하고 있다.

1. 서 론

오늘날의 분산 컴퓨팅 환경에서, 다양한 시스템의 관리를 위한 전문가의 필요성이 증가하고 있다. 그러나 이용 가능한 인적자원과 효과적인 비용관리의 측면에서 볼 때 인간에 의한 시스템의 관리는 명확한 제약이 있다[1]. 심지어는 전체 컴퓨터 시스템의 오류 중 약 40%가 관리자의 오류에 의한 것이라고 하니[2], 전문 관리자에 의존하는 현재의 시스템 관리 방식은 개선되어야 한다.

기존의 자가 치유 시스템은 Monitoring, Transaction, Analysis, Diagnosis, Feedback의 5-step의 프로세스를 가진다. 이 아키텍쳐는 리소스의 낭비, 로그의 수와 크기 증가, 관리자 및 벤더(Vender)의 의존성과 같은 몇 가지 결점을 포함하고 있다.

결과적으로, 본 논문에서는 위에서 언급한 문제들을 해결하기 위한 자가 치유 시스템을 제안한다. 제안 시스템은 다음과 같은 기능을 가진다. 1) 단일 프로세스 (Monitoring Module)의 사용을 통해 요구되는 리소스의 최소화 2) Alert situation, Emergency situation, Error situation 등등과 같은 컴포넌트의 상황에 따라 다양한 치유전략을 제공하는 Meta Policy의 사용 3) 신속하고 효과적인 자가 치유 시스템을 만족하기 위해, 우리는 Monitoring, Filtering, Translation, Analysis, Decision, Feedback 단계의 6-setp 프로세스를 사용한다.

제안 시스템은 실제로 프로토타입 형태의 실험을 위해 설계, 구현하였다. 섹션 2에서는 관련연구를 요약하였고, 섹션 3에서는 제안 시스템, 섹션4에서는 구현 및 평가를 기술하였고, 마지막으로 섹션 5에서는 결론을 서술하였다.

2. 관련연구

2.1 Self-Adaptive Software

자가 적응 소프트웨어는 자원의 변화와 시스템 오류 등에 대응하는 시스템 스스로의 행동을 수정하기 위한 능력을 가진다. 이들 자가 적응 소프트웨어는 자가 치유 시스템의 본질이다. 즉, 자가 치유 시스템은 self-adaptive behaviors을 포함해야 한다 [3]. Oreizy et. al. [4]는 self-adaptive software를 위해 아래와 같은 프로세스를 제안했다.

- Monitoring the system
- Planning the changes
- Deploying the change descriptions
- Enacting the changes

2.2 Adaptive Service Framework (ASF)

Adaptive Service Framework (ASF) [5] 는 IBM과 CISCO에 의해 제안되었으며 자가 적응형 행동의 형태를 가진다. ASF의 기능은 다음과 같다.

- Monitoring: *Adapter*[5]들이 다양한 컴포넌트들에 의해 생성되는 로그들을 모니터한다.
- Translation: 컴포넌트들이 로그를 발생시키면 Adapter들은 발생된 로그를 공통된 형태인 CBE (Common Base Event) [5]형태로 변환 시킨다.
- Analysis: Autonomic Manger [5]는 CBE로그를 분석하고 컴포넌트들의 연관관계를 식별하여 의존성을 파악한다.
- Decision and Feedback: the Autonomic Manager는 Symptom Rule[5]과 Policy Engine[5]에 의해서 적절한 치유방법을 결정하고, 해당 컴포넌트에 치유 방법을 수행한다. 치유 방법을 수행한 후 Resource Manager[5]는 치유 결과를 Autonomic Manager에게 피드백을 한다.
- 마지막으로 컴포넌트에 심각한 문제를 가진 경우 Call Home Format에 맞게 작성된 Message를 SSP (Support Service Provider) / Vendor에게 보내어 필요한 방법을 찾는다.

그러나, 기존 시스템은 아래와 같은 문제점이 있다.

- CBE로 변환된 로그의 사이즈가 변환되지 않은 로그의 사이즈보다 더 크다.(이 문제는 시스템의 성능을 저하 시킬 것 이다.

- 복잡한 연산으로 인하여 변환과정 중에 디스크, CPU, 메모리 사용량이 증가하는 문제점이 있다.
- ASF는 컴포넌트의 수만큼 많은 Adapter를 가지고 있기 때문에, 특히 유비쿼터스 환경에서 사용되는 handheld devices의 경우 불충분한 리소스의 문제를 야기할 수 있다.
- ASF는 위급한 상황에 대응하는 즉각적인 행동이 부족하기 때문에, 높은 치유시간을 소비 한다.

3. 제안시스템

제안 시스템의 구성은 그림 1과 같다. 제안 시스템은 다양한 경로의 컴포넌트들로부터 생성되는 로그를 모니터링하는 Monitoring Module, 생성된 로그를 Filtering하고 CBE 형식의 로그로 변환하는 Component Agent, 시스템의 상황을 모니터링하고 치유하는 System Agent, 치료받을 시스템의 상태를 진단하는 Diagnosis Agent, 진단된 결과에 대한 치유방법을 결정하는 Decision Agent, 특정 컴포넌트에 대한 벤더(Vendor)의 웹서버(Web Server)를 검색하여 필요한 정보를 가져오는 Searching Agent, 관리자에 치유를 위한 정보를 제공하고 스크립트를 작성하여 시스템의 치유를 위한 지식을 학습시키는 Admin Interface로 구성되어 있다.

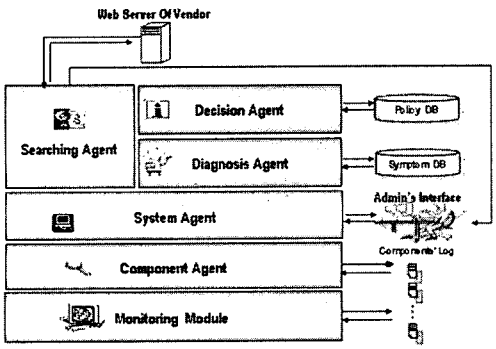


그림 [1] 제안 시스템 구성도

3.1 Monitoring Module

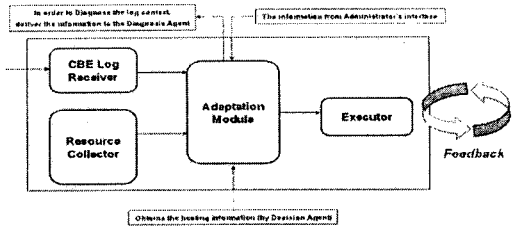
이 모듈은 각 컴포넌트들의 로그 생성을 실시간으로 모니터링 하기 위해 로그 경로에 대한 정보와 이 정보에 대응하여 로그를 CBE형식으로 변환하는 Component Agent에 대한 경로를 이차원 배열(Array) 내에 문자열(String)값으로 가지고 있다. Monitoring Module은 자신이 가지고 있는 모든 로그의 경로에 순차적으로 접근하여 로그파일의 사이즈가 변경되었는지 확인하고 사이즈 변경이 발생하면 대응되는 Component Agent를 실행시킨다.

3.2 Component Agent

Monitoring Module이Component Agent을 실행시키면 이 Module은 자가 치유(Self-healing)을 위해 필요한 'not', 'unused', 'error', 'reject' 와 같은 로그내용을 필터링하여 CBE 형식으로 변환한다. 이 과정을 위해 Component Agent는 Preprocessing, Parsing, XML Generating. 단계를 거친다. Preprocessing 단계를 거치면 하나의 로그는 작은 단어들로 나뉘게 된다. 이렇게 나눠진 단어들은 Parsing 과정을 통해서 CBE에서 의미하는 속성 값들과 매핑된다. 이 때 로그에 포함되어져 있지 않는 내용이 CBE log를 제작하기 위해 필요한 경우가 있다. 예를 들면 IP address, Host name etc. 이와 같은 값들은 미리 Configuration File에 지정해두고 필요할 때 사용할 수 있게 한다. 위의 과정을 통해 현재의 시스템의 상황이 Normal situation, Emergency situation, Error situation 한지를 판단하고, 만약 Emergency situation 이라면, Component Agent가 즉각적인 대응을 한다.

3.3 System Agent

System Agent은 다음 [그림 2]과 같이 여러 모듈로 나뉘어져 있다.



그림[2] System Agent

System Agent는 CBE Log Receiver, Resource Collector, Adaptation Module, Executor로 구성된다. CBE Log Receiver로부터 CBE로그가 들어오고, Resource Collector 은 Adaption Module에게 전달하기 위하여 CPU 정보, Memory 정보, process 정보, Job Schedule 정보를 수집한다. Adaptation Module은 수집된 시스템 정보와 CBE로그를 System Agent를 통해서 시스템의 현재 상태를 파악한다. 긴급한 상황이 아니면 Adaptation Module은 수집된 정보를 Diagnosis Module에게 전송한다. Executor는 Decision Agent로부터 받은 치유방법을 수행 한다.

3.4 Diagnosis Agent

System Agent로부터받은 로그들의 내용을 Symptom DB를 기반으로 각 로그간의 연관성을 파악하고, 상태를 진단한다.

3.5 Decision Agent

Diagnosis Agent로부터 받은 진단정보를 Policy DB를 기반으로 치유 방법을 결정하여, System Agent에게 전달하고, 그 결과를 피드백 받아 긍정적 피드백과 부정적 피드백을 나누어 가장 적절한 치유 방법을 결정한다.

3.6 Searching Agent

시스템의 문제를 해결하기 위한 진단 정보를 찾을 수 없을때, 벤더의 웹사이트를 검색하여 유용한 정보를 검색한다. 검색된 내용은 관리자에게 전달된다.

3.7 Administrator Interface

Searching Agent가 보내주는 정보를 볼 수 있는 웹브라우징 인터페이스와 다운로드한 패치파일을 설치할 수 있는 인터페이스를 제공한다. 사용자가 위의 인터페이스를 통해 작업한 내용은 스크립트로 작성되어지고, System Agent는 이 스크립트를 실행하여 결과를 피드백(Feedback)한다.

3.8 자가 치유를 위한 메타 정책

우리는 제안 시스템의 컴포넌트를 재구성하는 메타 정책을 기술 하였다. 메타 정책을 사용하여 agent들은 적절한 적응 정책(Adaptation Policy)을 선택할 수 있고, 그림 [3]은 메타 정책을 통해 Emergency 상황, Alert 상황 일때 적절한 전략이 선택되어지는 것을 볼 수 있다.

4. 구현 및 평가

4.1 시스템 구현

우리는 JAVA SDK1.4, Oracle 9i DBMS를 사용하였고, Agent 개발을 위하여 JADE1.3을 사용하였다. Apache Web server에서 생성된 로그 파일을 이용하였다. 그림 [4]는 Monitoring Module의 결과를 나타내었고, 그림 [5]는 Component Agent에 의해서 필터링된 결과를 나타내었다. 마지막으로 그림 [6]은 그 결과를 바탕으로

CBE로 변환된 것이다.

```
<MetaPolicy>
 <Agent name = "Component Agent">
  <Policy name = "Eermgency">
   <statusName = "no-operation"
       resourceName = "Component">
       method = "default" value = "terminated">
      urn : emergencyPolicy.xml
   </status>
  </Policy>
 </Agent>
 <Agent name = "System Agent">
  <Policy name = "Alert">
   <statusName = "" resourceName = "process">
    method = "processNum" value = "70">
    urn : AlertPolicy.xml
   <status>
  </Policy>
 </Agent>
  ...
</MetaPolicy>
-------------------------------------
<? xml version = "1.0" encoding = "UTF-8" ?>
<emergencyPolicy name = "Component Agent">
 <status name = "no-operation">
  <behavior strategy = "restart"
   method = "urn:restartOperation.sh">
 <status>
</emergencyPolicy>
```

그림[3] 자가 치유를 위한 메타 정책

```
The size of File 3507bytes
The command that can execute the agent:
java jade.Boot -host pjm -container
        sender:SHS_ComponentAgent.java
The location of log:
D:\RELATED_WORK\AUTONOMIC\PROJECT\S
elf-healing-System\SHS_Version05(MM)\BBB\error
```

그림[4] 모니터링 결과

```
date : [Fri Mar 17 11:32:42 2005]
log_level : [Alert]
client_ip : [client 203.252.53.142]
log_description :
httpd: Could not determine the server's fully
qualified domain name
```

그림[5] Component Agent에 의해서 필터링된 결과

```
<affectedComponentId componentAddressType="HostAddressType">
   <componentCore InstanceId="A001" applicationName="Apache" executionEnvironments="RedHatLinux">
   </componentCore>
   <componentType name="Web_Application_Server">
   </componentType>
   <componentAddress HostAddressType="pjm">
   </componentAddress>
</affectedComponentId>

<componentImpact>
   <componentCore InstanceId="A001" applicationName="Apache" executionEnvironments="RedHatLinux">
   </componentCore>
   <componentType name="Web_Application_Server">
   </componentType>
   <componentAddress HostAddressType="pjm">
   </componentAddress>
</componentImpact>
```

그림[6] CBE형식으로 변환된 로그

4.2 시스템 평가

시스템 평가는 로그 필터링, 메모리 사용량 및 변환과정에서의 로그 사이즈와 로그의 수 그리고 평균 치유 시간을 측정하여 평가하였다.

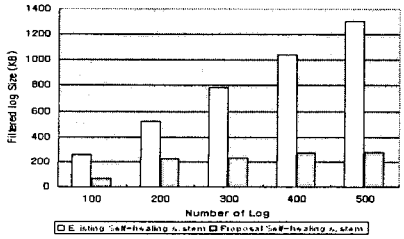


그림 [7] 로그의 수와 사이즈 비교

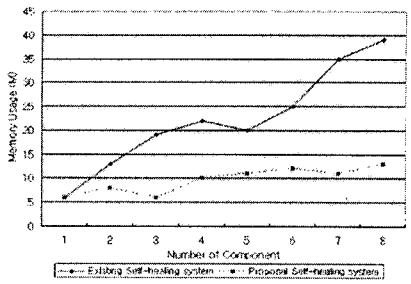


그림 [8] 메모리 사용량

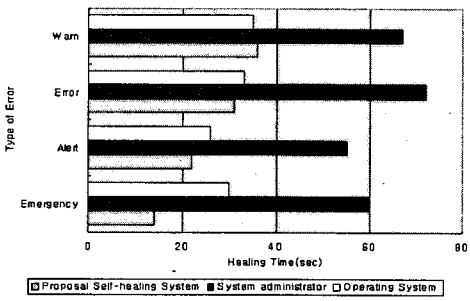


그림 [9] 치유 시간 측정

그림 [9]에서 보인 것 처럼, 우리는 시스템의 error의 타입을 분류하여 그것들에 대한 치유 시간을 측정한 결과 제안 시스템은 기존의 시스템보다 좋은 결과를 가져왔다.

5. 결론

본 논문에서는 Ubiquitous환경에서 여러 시스템들이 스스로 자신의 오류나 문제를 관찰, 진단하고 치유할 수 있는 자가 치유 시스템(Self-healing system)을 제안하여 기존의 관리자의 지식과 경험에 의한 시스템보다 우수함을 입증하였으며, 기존에 제안되었던 자가 치유 시스템(Self-healing system)보다 더 나은 성능을 보였지만, 그러나 좀더 다양한 상황에 대한(Context Awareness) 신속한 대처가 필요하고, 여전히 벤더와 관리자의 의존성이 존재하고, 문제점을 진단할 때 단순한 규칙 기반이 아닌 정확하면서도 광범위한 추론을 할 수 있는 알고리즘이 필요하다.

참고문헌

[1] http://www.ibm.com/autonomic

[2] IBM: Autonomic Computing: IBM's Perspective on the State of Information Technology, http://www-1.ibm.com/industries/government/doc/content/resource/thought/278606109.html.

[3] Hillman, J. and Warren, I. Meta-adaptation in Autonomic systems, In Proceedings of the 10th International Workshop on Future Trends in Distributed Computer Systems (FTDCS), Sozhou, China, May 26-28 2004

[4]. P. Oreizy et. al.: An Architecture-Based Approach to Self-Adaptive Software, IEEE Intelligent Systems, Vol. 14, No. 3, May/June (1999) 54-62.

[5] J. Baekelmans, P. Brittenham, T.Deckers, C.DeLaet, E.Merenda, BA. Miller, D.Ogle, B.Rajaraman, K.Sinclair, J. Sweitzer: Adaptive Services Framework CISCO white paper, October (2003)