자가 적응 모듈을 위한 목표 기반의 성능 개선 기법

이준훈○, 박정민, 이은석
성균관대학교 정보통신공학부
e-mail:{trsprs○, jmpark, eslee}@ece.skku.ac.kr

Goal-based Performance Improvement for Self-Adaptive Module

Joonhoon Lee○, Jeongmin Park, Eunseok Lee
Department of Information and Communication, Sunkyunkwan University

요 약

오늘날 컴퓨팅 환경은 점차 복잡해지고 있으며, 복잡한 환경을 관리하기 위해 많은 노력을 하고 있다. 이러한 관리를 좀 더 효율적으로 하기 위하여 환경에 스스로 적응하는 자가 치유에 관한 연구가 중요한 이슈가 되고 있다. 이러한 자가 치유를 하기 위해서는 추가적으로 리소스를 더 사용하게 된다. 우리의 이전 연구에서는 이러한 자가 적응 모듈이 사용하는 리소스를 줄여 성능을 향상시키고자 스위치를 이용하여 자가 적응 컴포넌트의 동작을 조절하는 방법을 제안하였다. 그러나 이러한 방법론은 자가 적응 모듈의 동작을 제어하기 위한 추상화(abstraction) 기법을 제공하지는 않는다. 또한 자가 적응 모듈을 설계할 때 개발자가 이 방법론을 적용하기 위한 코드를 직접 작성해야 한다. 본 연구에서는 이전 연구를 확장하여 1) 목표 그래프를 통해 자가 적응 모듈의 동작 단계를 분석하고 2) 기술된 단계를 이용하여 단계별 동작 스위치를 자동 생성한다. 이러한 방법론을 통하여 자가 적응을 위해 추가로 사용해야 하는 리소스의 사용을 줄일 수 있으며, 개발자가 자가 적응 모듈의 성능 개선을 위한 코드를 작성하는 수고를 덜 수 있다. 본 논문에서는 평가를 위하여 비디오 회의 시스템 내의 파일 전송 모듈의 목표 그래프를 작성하였다. 이 목표 그래프를 기반으로 자가 적응 모듈의 성능을 개선할 수 있는 동작 스위치의 템플릿 코드를 생성한다. 이러한 과정을 통해 생성된 코드를 자가 적응 모듈에 적용하여 스위치가 제대로 동작함을 확인한다. 또한 동작 스위치를 적용하기 전과 적용한 후의 동시 동작 컴포넌트 수를 비교한다. 이를 통해 목표 그래프를 기반으로 생성된 코드가 자가 적응 모듈의 성능을 향상시킬 수 있음을 확인할 수 있었다.

1. 서론

오늘날 컴퓨팅 시스템 환경이 갈수록 복잡해지면서 사람이 복잡한 시스템에서 발생할 수 있는 문제 상황들을 완벽하게 분석하여 시스템을 설계하는 것은 대단히 어렵다 [1]. 따라서 시스템의 문제를 발견하고 해결하기 위한 방법이 필요하다. 이러한 과정에서 시스템 관리자가 수동으로 새로운 어플리케이션을 실행하거나 서버의 구성을 변경하여 서버에 주어지는 부하를 줄이는 것과 같이 사람이 직접 시스템을 관리할 수 있다 [7]. 하지만 이러한 방법은 복잡한 단계를 요구하며 시스템 관리자가 능력에 따라 적응 여부가 다르다 [2]. 이러한 관점에서 관리를 위해 점차 늘어나는 비용을 줄이기 위해 시스템 스스로 리소스의 가변성이나 사용자의 요구 사항 또는 시스템의 오류를 진단하고 처리하는 자가 치유 시스템이 요구되고 있다 [3, 4].

시스템의 관리를 위한 자가 치유 시스템은 부가적인 모듈이 있어야 하므로 리소스를 더 많이 사용하게 된다. 우리의 이전 연구 [5]는 이러한 문제를 해결하기 위해 자가 적응 모듈이 사용하는 리소스를 줄이는 방법을 제안하였다. 그러나 이러한 연구는 자가 적응 모듈이 리소스를 줄이기 위한 추상화(abstraction) 기법을 제공하지 않는다. 또한 이러한 방법론을 적용하기 위해 개발자의 많은 노력이 필요하다.

본 연구에서는 이러한 문제를 해결하기 위해 다음과 같은 제안사항들을 제시한다.

- 자가 적응 모듈의 진행 단계를 나타내는 목표 그래프를 통해 모듈의 동작 단계를 분석한다.
- 동작 단계를 이용하여 단계별 자가 적응 컴포넌트의 동작 스위치를 자동 생성한다.

제안 방법론을 통해 자가 적응 모듈이 사용하는 리소스를 줄여, 시스템 스스로 문제를 해결하는 동시에 자가 적응 모듈로 인해 발생하는 시스템 성능의 감소를 줄일 수 있다.

본 논문에서는 평가를 위하여 간단한 비디오 회의 시

스템을 구현하였고, 시스템 내의 파일 전송을 위한 자가 적응 모듈을 구현하였다. 이 자가 적응 모듈의 동작 단계를 나타내는 목표 그래프를 그렸을 때, 자동으로 동작 스위치를 생성하여 자가 적응 모듈에 적용시켰다. 이를 통해 자가 적응 모듈에서 동시 동작하는 컴포넌트 수를 적용 전과 적용 후로 나누어 비교하였다.

본 논문의 구성은 다음과 같다. 2장에서는 관련 연구를 소개하고, 3장에서는 본 논문에서 제안하는 내용을 소개한다. 4장에서는 본 논문에서 제안하는 내용의 구현과 평가를 한다. 마지막으로 5장에서는 결론을 기술한다.

2. 관련 연구

본 장에서는 David S. Wile의 안전한 파일 전송 시스템 [2]와 성능 향상을 위한 동작 스위치[5]의 특징을 기술하고 장단점을 요약한다.

2.1 안전한 파일 전송 시스템

David S. Wile [2]은 파일 전송 시 의심스러운 행동을 탐지하고 대처하기 위해 네 가지 계층(Probe Layer, Gauge Layer, Controller Layer, Effector Layer)을 이용하였다. Probe 계층에서 대상 시스템을 모니터링 하고, Gauge 계층에서 수집된 정보들을 분석한다. Controller 계층에서는 Gauge 계층에서 분석된 정보들을 바탕으로 적응 판단을 한다. 이렇게 판단된 것은 Effector 계층에 의해 대상 시스템에 적용된다. 이를 통해서 대상 시스템의 내부를 수정하지 않고 외부에서 파일 전송 시 바이러스 등에 의해 의심스러운 행동(예, 포트 열기)을 차단할 수 있다.

그러나 이러한 과정에서 많은 컴포넌트들이 낭비된다. 다시 말해 현재 필요하지 않은 컴포넌트들도 실행을 계속하기 때문에 사용하지 않는 컴포넌트들이 리소스를 낭비하게 된다.

2.2 성능 향상을 위한 동작 스위치

[5]에서는 자가 적응 모듈에서의 단계를 나누어 각 단계별로 작동해야 하는 컴포넌트를 동작시키는 동작 스위치를 제안하였다. 동작 스위치에 의해 현재 사용하지 않는 컴포넌트를 동작하지 않도록 하여 현재 상태에서 불필요한 컴포넌트가 낭비하는 리소스 사용을 줄여 시스템의 성능에 영향을 덜 미치게 한다.

그러나 이러한 방법은 단계를 나누기 위한 추상화 기법이 없으며, 동작 스위치를 생성하기 위하여 개발자의 노력이 요구된다.

3. 제안 사항

본 논문에서는 앞서 언급한 문제점을 해결하기 위해, [7]에서 사용한 목표 그래프를 응용하여 자가 적응 모듈의 성능을 개선하는 방법론을 추상화할 수 있도록 한다.

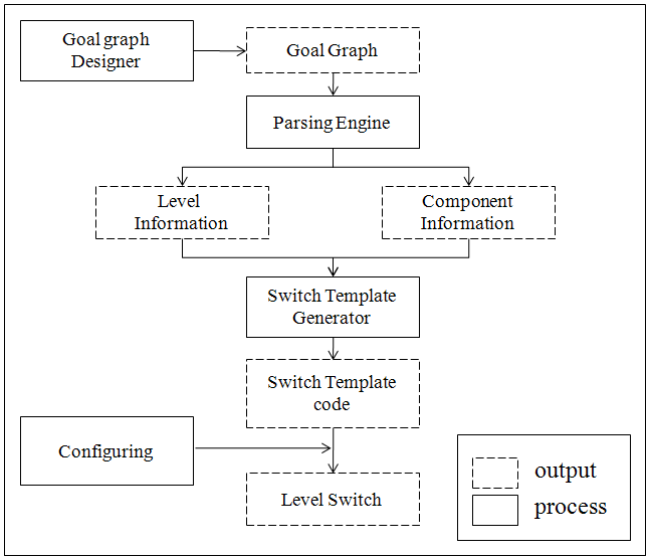
<표 1> 제안하는 컴포넌트의 기능 요약

이름	기능
Goal graph Designer	단계 인식을 위한 목표 그래프 작성
Parsing Engine	목표 그래프에서 필요한 정보 추출
Switch Template Generator	스위치 템플릿을 작성

3.1 제안하는 소프트웨어 구조

제안 소프트웨어 구조는 그림 1과 같이 구성하였다. 각 컴포넌트의 기능은 표 1에서 간략하게 설명한다. 이 구조는 다음과 같은 목적을 가진다.

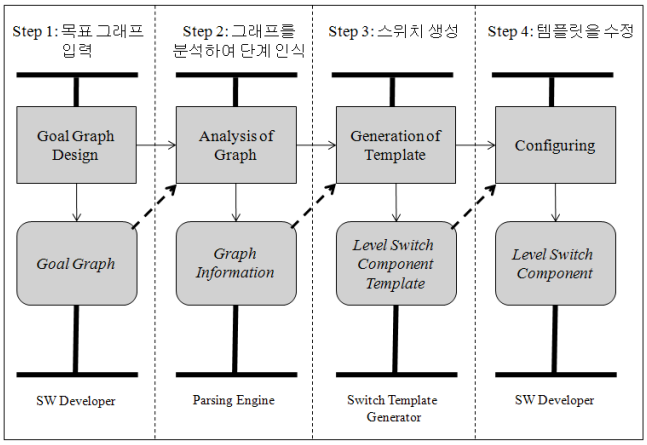
- 자가 적응 모듈의 단계를 시각화한다.
- 자가 적응 모듈의 성능 향상을 위한 동작 스위치를 자동 생성한다.



(그림 1) 제안하는 소프트웨어 구조

제안한 세부 컴포넌트들의 설명은 다음과 같다.

- Goal graph Designer: 개발자가 자가 적응 모듈의 단계를 나타내기 위한 목표 그래프를 그리는 도구이다. 목표 그래프는 표 2에 나타난 것과 같은 구조로 그릴 수 있다.
- Parsing Engine: 개발자에 의해 입력된 목표 그래프의 XML 파일을 분석하여 단계별로 작동해야 할 컴포넌트가 어떤 것인지를 인식하고, 그 정보를 저장한다.
- Switch Template Generator: Parsing Engine에 의해 작성된 정보들을 바탕으로 단계별 동작 스위치의 템플릿 코드를 생성한다.



(그림 2) 동작 스위치 생성 프로세스 (4-steps)

3.2 동작 스위치 생성 프로세스

위에서 언급한 소프트웨어 구조에서 동작 스위치를 생성하기 위한 프로세스는 그림 2와 같다. 1) 목표 그래프를 입력하고, 2) 그래프를 분석하여 단계를 인식한 후, 3) 동작 스위치를 위한 템플릿 코드를 생성하고, 4) 개발자가 템플릿을 수정하여 자가 적응 모듈의 성능을 개선하는 동작 스위치를 완성한다.

- 목표 그래프 입력 단계(step 1): 개발자는 Goal Graph Designer를 이용하여 대상이 되는 자가 적응 모듈의 목표 그래프를 그린다. 목표 그래프는 표 2에 나타난 구조를 이용하며, 그림 3에 그려진 것과 같이 그린다. 개발자에 의해 그려진 목표 그래프는 XML 파일로 저장된다.
- 그래프 분석 단계(step 2): Goal Graph Designer에 의해 생성된 XML 파일을 분석하여 단계를 나타내는 정보, 각 단계별로 해야 하는 작업, 각 작업별로 동작해야하는 컴포넌트들의 이름을 추출한다.
- 스위치 템플릿 생성 단계(step 3): Step 2에서 추출된 정보를 바탕으로 단계별 동작 스위치 템플릿 코드를 생성한다. 작성된 템플릿 코드에는 동작 단계별로 스위치를 ON, OFF 시키는 방법, 스위치의 상태에 따라 컴포넌트가 동작하는 방법이 포함된다.
- 템플릿 수정과 완성 단계(step 4): 이전 단계에서 생성된 템플릿 코드를 개발자에 의해서 수정한다.

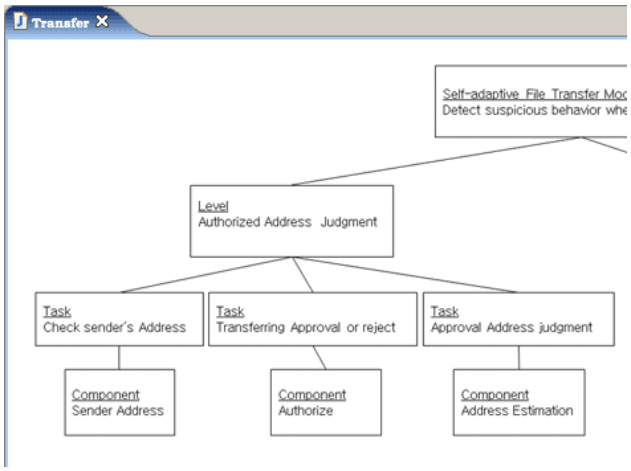
<표 2> 목표 그래프의 구조

Level	표시	설명
L0	Goal	모듈의 목표
L1	State	모듈의 작동 단계
L2	Task	각 단계의 세부 작업
L3	Component	세부 작업을 담당하는 컴포넌트

4. 구현 및 평가

화상 회의 시스템의 목적은 성공적인 회의를 하는 것이다. 회의가 진행되는 동안 사용자들은 다양한 종류의 문서들을 교환해야 한다. 이 과정에서 다수의 파일 송수신이 있다고 가정 한다. 본 논문에서는 제안 방법론을 화상 회의 시스템에 존재하는 파일 전송 모듈에 적용하여 목표를 기준으로 평가하는 것을 초점으로 한다.

화상 회의 시스템을 통해 사용자들이 회의를 진행하면서 파일 전송을 한다. 테스트를 위하여 화상 회의 시스템을 간단히 구현하여 이 시스템에 제안 사항을 적용하였다. 서버는 Java로 작성하였으며, JDK 1.5를 이용하였다. 서버에서 필요한 모니터링 도구들과 클라이언트는 C#을 이용하였다. 영상 처리를 위하여 클라이언트는 DirectShow를 추가로 이용하였다.



(그림 3) 목표 그래프 작성 플러그인

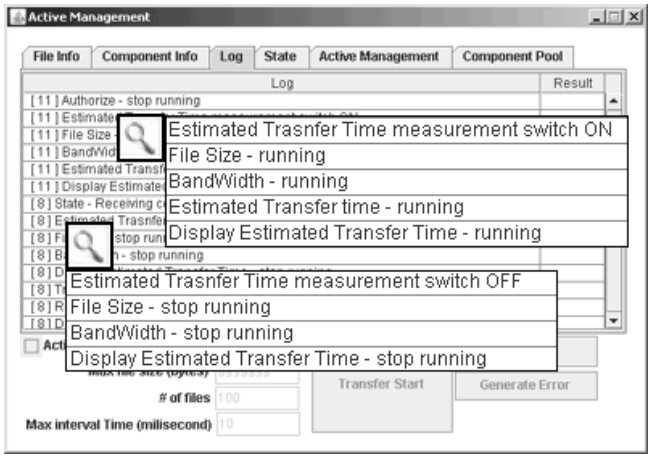
목표 그래프를 그리기 위하여 IBM에서 개발한 eclipse 3.2의 플러그인 개발 환경(Plug-in Development Environment)과 GMF(Graphic Modeling Framework)를 이용하여 그림 3과 같은 플러그인을 구성하였다. 이 플러그인을 통해 그려진 그래프는 XML로 저장되어 다른 컴포넌트들에서 사용할 수 있도록 하였다. 제안 방법론을 통해 생성된 코드는 그림 4와 같다.

```
...

/**
 * To turn off the switch of this component
 *
 */
public synchronized void turnOff() {
 checkWait = true;
 ComponentView.setSwitch(num, checkWait);
}

/**
 * To turn on the switch of this component
 *
 */
public synchronized void turnOn() {
 notify();
 checkWait = false;
 ComponentView.setSwitch(num, checkWait);
}
```

(그림 4) Template Generator가 생성한 코드 예제 (Java)

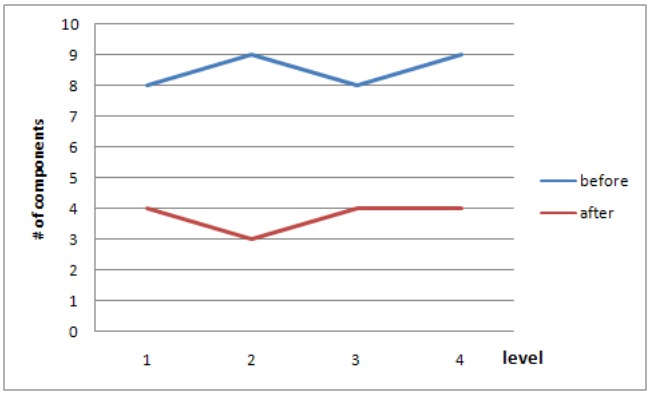
정상적인 경우, 목표 그래프에 나타난 단계별로 스위치

가 동작하여 해당 컴포넌트들을 동작 시킨다. 다음 단계로 넘어가면 현재 작동 중이던 컴포넌트를 중지하고 다음 단계의 컴포넌트를 동작시킨다. 그림 5는 이러한 과정을 로그로 표시한 것이다.



(그림 5) 동작 스위치의 정상적인 작동을 나타내는 로그

평가를 위하여 자가 적응 모듈이 동작할 때의 각 단계별로 동작하고 있는 컴포넌트의 개수를 비교하였다. 제안 방법론을 적용하기 전의 컴포넌트 수에 비해 적용한 후의 컴포넌트 수가 더 줄어들었다. 이를 통해 시스템에서 자가 적응 모듈이 사용하는 리소스의 양을 줄여 시스템 성능에 자가 적응 모듈이 미치는 영향을 최소화할 수 있음을 확인할 수 있었다. 본 실험의 결과는 그림 6에 나타나 있다.



(그림 6) 동시 동작하는 컴포넌트 수

5. 결론

본 연구에서는 자가 적응 모듈이 시스템의 성능에 영향을 주는 부분을 최소화 하는 추상화 기법을 제안하였다. 제안 방법론을 통해 다음과 같은 이점을 얻을 수 있다.

- 자가 적응 모듈의 동작 단계를 시각화할 수 있다.
- 자가 적응 모듈의 성능 개선을 위한 코드를 작성하기 위한 수고를 덜 수 있다.

자가 치유 연구에서는 여러 모듈을 추가로 사용하기 때문에 시스템의 리소스를 사용하게 되고 시스템의 성능을 필연적으로 감소시킨다. 본 논문에서는 자가 적응 모듈의 성능 개선을 위한 방법 중의 하나인 동시 동작 컴포넌트 수를 줄이는 방법을 목표 그래프를 통해 자동 생성할 수 있음을 확인하였다. 그러나 단계를 구분할 수 없는 자가 적응 모듈에는 이러한 방법론을 적용할 수 없다는 문제가 존재한다. 또한 개발자가 그린 목표 그래프가 제대로 그려졌는지에 대한 검증과 자가 적응 모듈의 성능 향상을 위한 다른 방법에 대한 연구는 향후 과제로 남겨 둔다.

참고문헌

[1] Jiwen Wang, Chenghao Guo and Fengyu Liu, “Self-healing Based Software Architecture Modeling and Analysis Through a Case Study,” Proceedings of Networking, Sensing and Control, pp.873-877, 2005.

[2] David S. Wile and Alexander Egyed, “An Externalized Infrastructure for Self-Healing Systems,” Proceedings of the 4th Working IEEE/IFIP Conference on Software Architecture, pp.285-288, Sep., 2004.

[3] Robert Laddaga, “Creating robust software through self-adaptation,” Proceedings of Intelligent Systems and Their Applications, Vol. 14, Issue 3, pp.26-29, 1999.

[4] Jeffrey O. Kephart, David M. Chess, “The Vision of Autonomic Computing,” IEEE Computer Society, Vol. 36, pp.41-50, Jan., 2004.

[5] 최윤규, 박정민, 유길종, 이은석, “외장형 자가 적응 인프라스트럭처를 위한 제어 모듈의 성능개선,” 제 26회 한국정보처리학회 추계학술발표대회 논문집 제 13권 제 2호, pp.1149-1152, 2006.

[6] P. Stelling, I. Foster, C. Kesselman, C. Lee, and G. V. Laszewski, “A fault detection service for wide area distributed computations,” Proceedings of 7th High Performance Distributed Computing, pp.268-278, Jul., 1998.